МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ХАРЬКОВСКАЯ НАЦИОНАЛЬНАЯ АКАДЕМИЯ ГОРОДСКОГО ХОЗЯЙСТВА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения лабораторных работ по курсу «Информатика и компьютерная техника»

Visual Basic for Application

(для студентов 1, 2 курсов дневной формы обучения бакалавров направления 6.030504 – «Экономика предприятий», 6.030509 – «Учет и аудит»)

ХАРЬКОВ – ХНАГХ – 2008

Методические указания для выполнения лабораторных работ по курсу «Экономическая информатика» (для студентов 1, 2 курсов дневной формы обучения бакалавров направления 6.030504 – «Экономика предприятий», 6.030509 – «Учет и аудит») / Сост.: Б.И. Погребняк, А.В. Белогурова, Е.В Кузьмичева, Н.О. Манакова. – Харьков: ХНАГХ, 2008. – 86 с.

Составители: Б.И. Погребняк, А.В. Белогурова,
Н.О. Манакова, Е.В Кузьмичева,

Рекомендовано кафедрой *"Прикладной математики и информационных технологий",*
протокол № 9 от 1 февраля 2008г.

Оглавление

Введение

Самый первый вариант языка программирования BASIC (Бейсик) был создан в 1964 году сотрудниками Дартмутского колледжа Дж. Кемени и Т. Курцем. История происхождения названия языка тоже весьма интересна. В прошлом веке один английский миссионер выделил из английского языка триста наиболее употребительных слов, назвал их Basic English и стал обучать местных жителей. Опыт оказался весьма успешным, и контакты с ними значительно упростились. Создатели языка BASIC стремились достигнуть того же эффекта – облегчить взаимопонимание между начинающими программистами и компьютером. Аббревиатура BASIC так и расшифровывается – «Beginner’s All purpose Symbolic Instruction Code», что в переводе значит «многоцелевой язык символьных команд для начинающих».

Язык программирования VBA – Visual Basic for Applications, является результатом дальнейшего развития языка BASIC. Несмотря на то, что VBA для каждого из приложений, которое его поддерживает (Microsoft Excel, Microsoft Word и т.д.), имеет свой, особенный, набор возможностей, общая основа у него остается неизменной. То есть, освоив однажды VBA, например, для Microsoft Excel, использование его, затем, например, для автоматизации Microsoft Word не составит особого труда.

Лабораторная работа №1

«Создание и выполнение макросов»

Изучим: редактор VBA,запись и сохранение макросов, написание нового модуля процедуры, операторы вывода.

Как записать новый макрос

Перед тем как редактировать макрос с помощью VBA, вам нужно сначала его записать. Рассмотрим основные понятия, необходимые для того, чтобы научиться записывать макросы.

Как правило, запись макроса состоит из четырех шагов:

1. *Создание начальных условий для макроса* подразумевает приведение вашей программной среды в точно такое состояние, в котором вы предполагаете выполнять записываемый макрос.
2. *Запустить запись макроса и задать ему имя.* Как только вы запустите запись макроса, вы должны будете дать ему подходящее имя и указать, где сохранить этот макрос. Записывая макрос в программе Excel 2000, вы можете также назначить ему комбинацию клавиш.
3. *Выполнить действия, которые вы хотите записать.* Могут быть записаны любые действия, которые вы выполняете с помощью мыши или клавиатуры, включая выполнение ранее записанных макросов. Что именно вы запишете, зависит от того, для какого задания вы предполагаете использовать данный макрос.
4. *Остановить запись макроса.* После остановки записи ваши действия уже не фиксируются. Новый макрос готов к применению сразу после остановки записи.

Выполним последовательно все четыре шага.

Перед тем как записывать макрос, вы должны создать точно такие же условия, в каких потом вы будете его выполнять. Предположим, например, что вы хотите создать макрос, который будет задавать шрифт, кегль и цвет текста в выделенных ячейках листа Excel. Начальными условиями для такого макроса будет открытый лист Excel с выделенной одной или несколькими ячейками.

Вы обязательно должны создать начальные условия перед началом записи, потому что, если вы начнете запись, а потом откроете лист и станете выделять ячейки, все эти действия будут записаны в макросе. В результате ваш макрос будет уж очень специфичен — он всегда будет открывать одну и ту же книгу, один и тот же лист и форматировать в нем одни и те же ячейки. Для того чтобы создать универсальный макрос для форматирования клеток, вам нужно открыть книгу, выбрать в ней лист и выделить ячейки до начала записи.

1. Создайте начальные условия для записи макроса. Для этого запустите Excel, откройте новую книгу и заполните ячейки B4:D8

произвольным текстом и формулами, а в ячейку B2 введите свою фамилию.

Для того чтобы начать запись макроса в программе Excel 2000, выберите в меню Сервис-Макрос-Начать запись. Excel раскроет диалоговое окно Запись макроса. В этом окне вы должны задать имя макроса и указать, где его сохранить. Указать имя и место сохранения нужно до того, как вы начнете запись макроса. В этом же окне вы можете задать комбинацию клавиш, которой макрос будет запускаться на выполнение.

В диалоговом окне Запись макроса есть четыре управляющих элемента.

- *Текстовое поле* Имя макроса. Это первый параметр, который вам нужно указать в этом диалоговом окне. По умолчанию VBA предлагает вам имя, состоящее из слова Macro с последующей цифрой, соответствующей порядковому номеру создаваемого в этом сеансе работы макроса. Лучше заменить это имя на какое-нибудь осмысленное. Например, если вы создаете макрос, строящий диаграмму продаж на основании данных текущего листа, можете назвать этот макрос Диагр_продаж.
- *Текстовое поле* Сочетание клавиш. В этом поле вы можете задать комбинацию клавиш, с помощью которой ваш макрос будет запускаться на выполнение. Просто введите в поле подходящий символ на клавиатуре. Этим есть смысл пользоваться только в том случае, если вы собираетесь применять свой макрос часто. Все комбинации клавиш для вызова макросов в программе Excel состоят из клавиши <Ctrl> и какого-нибудь символа. Если вы введете в этом поле букву <a>, комбинацией для вызова макроса будет <Ctrl+a>, если вы введете <A>, комбинацией будет <Ctrl+Shift+A>.
- *Список* Сохранить в. В этом списке вы указываете, где должен быть сохранен новый макрос. Возможные варианты таковы: Личная книга макросов, Новая книга, Эта книга. Если вы выберете Личная книга макросов, ваш новый макрос будет записан в специальной книге с названием Personal.xls, которая открывается при каждом запуске программы Excel. Такой выбор уместен тогда, когда вы хотите, чтобы новый макрос был доступен при любом сеансе работы с Excel. Если вы выберете Эта книга, ваш макрос запишется в текущей открытой книге. Этот вариант годится, если вы хотите, чтобы макрос был доступен только при работе с этой книгой. Выбрав Новая книга, вы заставите Excel создать новую книгу и сохранить макрос в ней. При этом активной останется та книга, которая была у вас открыта перед записью макроса, и все ваши действия будут применены именно к ней, а не к новой книге. Независимо от того, в какой книге вы решите сохранить макрос, он будет записан в ней в виде *модуля*.
- *Текстовое поле* Описание. Информация в этом поле макросом не используется, а служит для вас и тех, кто будет пользоваться вашим макросом. Здесь очень полезно указать, для чего макрос предназначен и что

он делает. В начале записи Excel помещает сюда текущую дату и имя пользователя, которое указано на вкладке Общие диалогового окна Параметры в меню Сервис.

2. Выберите в меню Сервис-Макрос-Начать запись. Excel раскроет диалоговое окно В текстовом поле Имя макроса введите в качестве имени макроса Form12. Оставьте без изменений текст, который Excel вставила в поле Описание, но добавьте следующее: Форматирует текст в ячейке шрифтом Arial, полужирный, 12 пунктов. В списке Сохранить в укажите, где должен быть сохранен макрос. Выберите Эта книга. Если вы уверены, что часто будете пользоваться этим макросом, можете присвоить ему комбинацию клавиш для быстрого вызова. В таком случае введите клавишу в поле Сочетание клавиш диалогового окна Запись макроса. И запишите эту комбинацию в тетрадь. Щелкните на кнопке ОК.

Запустится программа записи макроса и откроется панель «Остановить запись».

(сейчас запись не останавливать)

3. Выделите диапазон ячеек B4:D8. Откройте диалоговое окно Формат ячеек и установите шрифт Arial, полужирный, 12 пунктов. Закройте окно, щелкнув ОК. Закончите запись макроса, щелкнув на кнопке Остановить запись

4. Перейдите на Лист2 и заполните ячейки B4:D8 произвольной информацией.

5. Запустите макрос Form12, выбрав в меню Сервис – Макрос – Макросы, выбрав имя макроса и щелкнув на кнопке Выполнить. Что вы наблюдаете?

Знакомство с редактором Visual Basic

Для того чтобы узнать, какие модули записаны в определенной книге и посмотреть текст этих модулей, вам понадобится редактор языка Visual Basic. Редактор Visual Basic — это инструмент для создания модулей и просмотра их содержимого, создания и редактирования текста макросов, создания диалоговых окон и решения других задач, с которыми вам приходится сталкиваться при работе с программами на языке Visual Basic. В дальнейшем мы будем называть этот редактор просто VB-редактором.

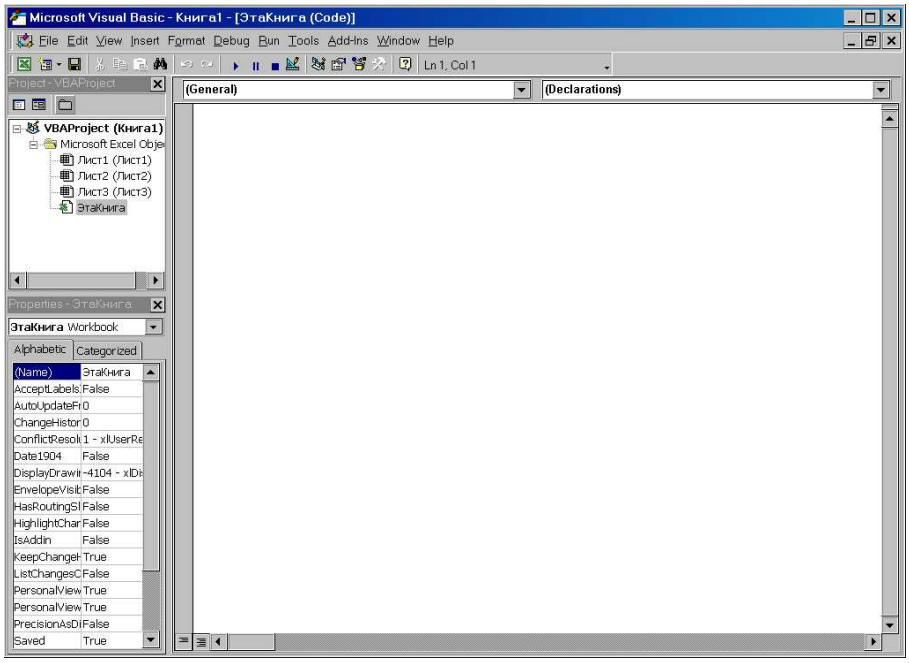
VB-редактор приложения Excel совершенно аналогичен VB-редакторам других приложений, поддерживающих язык VBA. Фактически, работая с Excel, вы пользуетесь тем же самым VB-редактором, что и при работе с Microsoft Word, Access и другими приложениями Office. Рассмотрим, как запустить редактор Visual Basic, объясним назначение его окон и команд меню.

Запускаем редактор VB

Сделать это можно двумя способами.

- Выберите в меню Сервис-Макрос-Редактор Visual Basic.
- Нажмите <Alt+Fll>.

Какой бы способ вы ни выбрали, запустится редактор Visual Basic. Рассмотрите окно этого редактора.



Окно редактора Visual Basic содержит три дочерних окна. Каждое из дочерних окон предоставляет некоторую информацию о вашем проекте. (Проектом называются модули и другие объекты, сохраненные в рабочей книге Excel или в шаблоне книги.) Обычно дочерние окна размещены у границы родительского окна *(docked position)* — правой, левой, верхней или нижней. По умолчанию дочерние окна редактора Visual Basic расположены у левой границы.

Приведем описание окон редактора Visual Basic.

- Окно Project (Проект). В окне Project вы видите древообразную структуру, описывающую ваш проект: открытые в данный момент книги, объекты, содержащиеся в этих книгах (модули, ссылки, формы и т.д.). Это окно предназначено для поиска модулей и других объектов в вашем проекте.
- Окно Properties (Свойства). В окне Properties приведены свойства выбранного в данный момент объекта. Иногда свойства объекта состоят только из его имени. На вкладке Alphabetic (По алфавиту) объекты приведены в

алфавитном порядке. На вкладке Categorized (По категориям) объекты отсортированы по категориям.

- Окно Code (Программа). В окне Code вы видите текст макросов. Именно в этом окне вы будете читать, создавать и редактировать свои макросы.

По умолчанию окно Project выглядит подобно окну программы Explorer и отображает папки вашего проекта. Щелчок на знаке "+" слева от элемента разворачивает данную ветвь дерева, а щелчок на знаке "—" сворачивает эту ветвь.

Для того чтобы выбрать объект в окне Project, нужно щелкнуть на этом объекте. В режиме отображения объекты сгруппированы в папках в соответствии с их типом — объекты Excel, модули и ссылки. Для переключения режимов отображения щелкните на кнопке Toggle Folders (Папки). В окне Project, помимо кнопки Toggle Folders, есть еще две кнопки: View Code (Программа) и View Object (Объект). Кнопка View Object отображает объект, выбранный в окне Project, а кнопка View Code отображает текст модуля в окне Code.

Рассмотрите окно Code, размер которого увеличен по сравнению со стандартным так, чтобы можно было разместить в нем как можно больше текста и лучше были видны различные его свойства. Вы видите текст макросов, записанных вами.

По умолчанию модуль отображается в режиме Full Module View (Представление полного модуля). В этом режиме просмотра в окне отображается весь модуль со всеми входящими в него макросами. Каждый макрос отделяется от соседних тонкой чертой. Редактор Visual Basic также позволяет вам просматривать модуль в режиме Procedure View (Представление процедуры). Макрос иногда называют процедурой. Режим просмотра можно менять, щелкая на кнопках в левом нижнем углу окна Code.

6. Запустите редактор VBA и рассмотрите открывшиеся окна. В окне Code выведите текст вашего макроса Form12. Переключите режим на Procedure View. Что изменилось?

7. Рассмотрите стандартную панель инструментов редактора VBA. Законспектируйте в тетради назначение каждой кнопки.

Панели инструментов редактора Visual Basic

Большинству пользователей легче щелкнуть мышкой на кнопке, чем выбрать команду из меню. Поэтому в редакторе Visual Basic для наиболее часто используемых команд предусмотрены кнопки на панели инструментов. Если вы интенсивно работаете с Visual Basic, то наверняка заметили, что кнопки намного ускоряют работу.

Обычно Visual Basic отображает только одну панель инструментов — стандартную. Но, помимо стандартной панели инструментов, есть еще три панели — Edit, Debug и UserForm.

На панели инструментов Edit имеется несколько кнопок, которые могут очень пригодиться при работе с исходным текстом макроса в окне Code. Более

того, на этой панели есть даже несколько кнопок, которых вы не найдете в меню Edit.

Для того чтобы выбрать, какую панель инструментов отображать, а какую нет, вы должны воспользоваться командой меню View-Toolbars. Поскольку панель Edit по умолчанию не отображается, вам придется вывести ее на экран самостоятельно. Может случиться так, что кто-то уберет с экрана стандартную панель инструментов, тогда вам вручную придется восстановить на месте и ее. Для того чтобы убрать или вывести на экран какую-то панель инструментов, сделайте следующее.

1. Выберите в меню View-Toolbars. При этом вы увидите дополнительное меню со списком всех доступных панелей инструментов.
2. Щелкните на имени нужной панели инструментов. Например, щелкните на строке Edit. Visual Basic отобразит выбранную панель.

Обычно редактор Visual Basic отображает стандартную панель инструментов прижатой к верхней границе окна. Но вы можете расположить любую панель инструментов где угодно, в любом месте окна. Такую панель инструментов называют *плавающей.* Она выглядит как полноценное окно, имеет границы и строку заголовка. В редакторе Visual Basic с плавающими панелями обращаются точно так же, как в программе Excel или в любой другой — ее можно перетаскивать, менять размеры и форму.

Сейчас мы рассмотрим команды стандартной панели инструментов, а позже — команды панели Edit.

Стандартная панель инструментов. На стандартной панели инструментов в редакторе Visual Basic имеется 18 кнопок. Каждой кнопке соответствует некоторая команда меню. Изучая таблицу, смотрите на окно редактора, чтобы понять, какой рисунок на кнопке соответствует той или иной команде.

Кнопки стандартной панели инструментов.

Кнопка	*Действие*
View (Вид) <приложение>	Переключает вас на исходное приложение, из которого был вызван редактор Visual Basic. Рисунок на этой кнопке зави сит от того, из какого приложения вы запускали Visual Basic.
Insert (Вставить) <объект>	Щелкните на стрелке справа от кнопки, и вы увидите - спи сок объектов, которые можно вставить в ваш проект.Это объекты UserForm, Module, Class Module

	(Модуль класса) или Procedure. Действует аналогично меню Insert.
Save (Сохранить)	Сохраняет текущий проект. Действует аналогично меню File-Save
Cut (Вырезать)	Вырезает выделенный текст и помешает его в буфер обмена. Действует аналогично меню Edit-Cut
Copy (Копировать)	Копирует выделенный текст и помещает его в буфер обмена. Действует аналогично меню Edit-Copy
Paste (Вставить)	Вставляет текст из буфера обмена в позицию курсора.Действует аналогично меню Edit-Paste
Find (Найти)	Раскрывает диалоговое окно для поиска текста. Действует аналогично меню Edit-Find
Undo (Отменить)	Отменяет последнюю команду. Отменить можно не всякую команду. Действует аналогично меню Edit-Undo
Redo (Повторить)	Повторяет последнюю выполненную команду. Действует аналогично меню Edit-Redo
Run (Запуск)	Запускает макрос, который вы сейчас редактируете, т.е. тот макрос, в тексте которого находится текстовый курсор. Действует аналогично меню Run-Sub/UserForm
Break (Прервать)	Прерывается выполнение макроса. Действует аналогично меню Run-Break
Reset (Сброс)	Сбрасывает значения всех переменных модуля и очищает стек вызовов. Действует аналогично меню Run-Reset
Design Mode	Включает или отключает режим конструктора. Действует (Конструктор) аналогично меню Run-Design Mode
Project Explorer	Раскрывает окно Project. Действует аналогично меню View-(Окно проекта) Project Explorer
Properties Window	Раскрывает окно Properties. Действует аналогично меню (Окно свойств) View-Properties Window
Object Browser	Раскрывает окно Object Browser. Действует аналогично меню - (Просмотр объектов) View-Object Browser
Toolbox (Панель)	Раскрывает панель инструментов Toolbox. Действует аналогично меню – (Просмотр элементов) View-Toolbox
Office Assistant	Вызывает окно справки. Действует аналогично меню Help-(Помощник по Office) Microsoft Visual Basic Help
Cursor Position	Эта область на панели не является командной кнопкой.

(Позиция курсора)	Здесь просто указывается позиция курсора в текстовом окне Code, а именно номер строки и столбца
More Buttons (Дополнительные кнопки)	Щелкнув на стрелке справа от кнопки, вы увидите дополнительное меню с командами, для которых можно создать кнопки на панели инструментов

8. Отредактируйте созданный макрос, добавив построчные комментарии и свою фамилию, группу и курс.

Редактируем макрос

Для того, чтобы начать редактировать макрос, вы должны сначала открыть модуль, содержащий этот макрос, и найти в модуле текст макроса. Например, для того, чтобы редактировать макрос Forml2, который вы записали, вы должны сначала открыть модуль, в котором хранится исходный текст макроса Forml2. Для этого выполните следующее.

1. Откройте редактор Visual Basic, нажав <Alt+Fll>, если он еще не открыт.
2. Раскройте окно Project, если оно еще неоткрыто. Для этого выберите в меню View-Project.
3. Найдите в древообразном списке окна Project модуль, который вы хотите отобразить. Если в окне Project включен режим отображения Folders, вы найдете все модули своего проекта в папке Modules.
4. Дважды щелкните на нужном модуле. При этом его текст раскроется в окне Code.

После этого вы можете воспользоваться списком Procedure, в котором вы найдете нужный макрос.

Для того чтобы добавить к тексту макроса комментарии, сделайте следующее.

1. Установите курсор в конец последней строки стандартного комментария.
2. Вставьте новую строку, нажав клавишу <Enter>.
3. Введите символ апострофа (') — все комментарии начинаются с этого знака.
4. Введите текст комментария.
5. Сохраните измененный макрос.

Вы должны получить код, аналогичный приведенному ниже.

```
Sub Макрос2()
" Макрос2 Макрос
' Макрос записан 05.02.2003 (ps)
'  Выполнил студент 2 курса Иванов Андрей
'  факультет ЭиМ
'    Выбор диапазона
   Range("B3:D8").Select
   ' Установка параметров шрифта
```

```
  With Selection.Font
    .Name = "Arial"
    .FontStyle = "полужирный"
    .Size = 12
    .Strikethrough = False
    .Superscript = False
    .Subscript = False
    .OutlineFont = False
    .Shadow = False
    .Underline = xlUnderlineStyleNone
    .ColorIndex = xlAutomatic
  End With
End Sub
```

9. С помощью редактора VBA напишите процедуру, которая будет выводить на экран окно приветствия “Hello, World!”. Сохраните ее под именем Hello.

Пишем текст процедуры

Перед тем как начинать запись текста процедуры в модуль, будь то новый или существовавший ранее модуль, вы должны поместить курсор в окно Code в то место, куда хотите вставить текст процедуры.

Вы можете вводить текст процедуры в любое место в модуле, лишь бы это было после слова End Sub, которое заканчивает предыдущую процедуру, и перед словом Sub, которое начинает следующую процедуру. Большинство программистов помещают новую процедуру в конец модуля.

Когда вы пишете процедуру, вы должны прежде всего указать ее имя и поместить в текст слова Sub в начале процедуры и End Sub в конце. Если хоть одно из этих требований не будет соблюдено, синтаксические правила будут нарушены и редактор Visual Basic выдаст сообщение об ошибке при попытке запустить процедуру.

В любом языке программирования классическим примером простой программы является программа, выводящая на экран слова **Hello, World!** В листинге вы видите такую программу, состоящую из единственной процедуры.

```
Sub hello()
MsgBox "Hello, World!", , "Окно вывода сообщений на экран"
End Sub
```

Для того, чтобы ввести такую *программу* самостоятельно, сделайте следующее.

- Убедитесь, что курсор находится в начале первой строки в окне Code и введите текст из листинга. В конце каждой строки нажимайте <Enter>, начиная новую строку.

- Вводите текст точно так же, как он приведен в листинге
- Переименуйте новый модуль, дав ему содержательное имя.

Visual Basic во многом помогает вам писать программу. Например, после того как вы введете строку, содержащую слово Sub и имя процедуры, он автоматически введет строку со словами End Sub. Таким образом, уменьшается вероятность того, что вы забудете ввести нечто важное.

Кроме того, редактор Visual Basic имеет свойство, называемое Auto Quick Info. Как только вы введете MsgBox и нажмете пробел, появится вспомогательное окно со списком аргументов функции, имя которой вы только что ввели, в данном случае функции MsgBox. Аргумент, значение которого вы должны сейчас вводить, выделяется полужирным шрифтом. Вспомогательное окно Quick Info закрывается, как только вы переместите курсор на другую строку. Его можно закрыть также, нажав клавишу <Esc>.

Первая строка в листинге — это объявление процедуры. Объявление процедуры служит для того, чтобы сообщить Visual Basic о существовании процедуры и о месте начала ее исходного текста. Каждое объявление процедуры состоит из слова Sub, пробела и имени процедуры. В листинге имя процедуры — Hello. Заканчивается объявление процедуры парой пустых скобок. Эти скобки являются обязательным элементом. Если вы не поставите скобок, Visual Basic сделает это за вас, когда вы нажмете <Enter> в конце строки.

Вторая строка в листинге составляет тело процедуры и является единственной инструкцией, которая что-то делает. Тело процедуры может состоять из нескольких инструкций. Инструкция MsgBox выводит на экран окно с сообщением пользователю "Hello, World!". Окно озаглавлено "Окно вывода сообщений на экран".

Третья и последняя строка в нашей процедуре — инструкция End Sub. Она служит для того, чтобы указать VBA конец процедуры. VBA, встретив такую инструкцию, заканчивает выполнение. Как и объявление процедуры, инструкция End Sub должна стоять в начале строки и быть единственной инструкцией в строке, но вы можете ставить комментарии после нее. Как уже говорилось, Visual Basic автоматически добавляет эту инструкцию в конце процедуры.

10. Запустите процедуру на выполнение.

1. Откройте окно Macro, выбрав в меню Tools-Macros.
2. В списке Macro Name выберите процедуру Hello.
3. Щелкните на кнопке Run.

В результате выполнения процедуры Visual Basic выведет окно приветствия. Щелкните на кнопке ОК и закройте окно.

11. Закройте редатор, сохраните книгу Excel на диске D/курс 2/Ваша группа/FIO.xls, где FIO – начальные буквы вашей фамилии, имени, отчества. Выйдите из Excel.

12. Запишите макросNewFIO (где FIO – начальные буквы вашей фамилии, имени, отчества), который будет открывать новую книгу и

затем сохранять ее под именем NEWFIO в вашем каталоге и закрывать ее. Запустите его на выполнение 2 раза. Результаты объясните и запишите в тетради.

13. Напишите процедуру, которая будет выводить сообщение «Привет от Excel!». Окно сообщения должно иметь заголовок «Сообщение Excel».

Варианты заданий

В текущей рабочей книге создать макрос, который:

1. Создаёт новую рабочую книгу с именем, указанным пользователем.
2. Добавляет в текущую рабочую книгу лист с именем, указанным пользователем.
3. Устанавливает в активной ячейке шрифт размером в 14 пунктов.
4. Устанавливает в активной ячейке полужирное начертание шрифта.
5. Устанавливает в активной ячейке курсивное начертание шрифта.
6. Устанавливает в активной ячейке шрифт Arial.
7. Записывает в активную ячейку Ваше имя.
8. Устанавливает вокруг активной ячейки широкую рамку.
9. Записывает в активную ячейку формулу =A1 + A2.
10. Открывает указанную пользователем рабочую книгу.
11. Делает активным, указанный пользователем, рабочий лист.
12. Делает активной, указанную пользователем, ячейку.
13. Выравнивает содержимое активной ячейки по центру.
14. Устанавливает в активной ячейке красный цвет шрифта.
15. Устанавливает в активной ячейке в зелёный цвет заливки.
16. Устанавливает указанное пользователем имя для выделенного рабочего листа.
17. Объединяет две смежные ячейки.
18. Копирует содержимое указанной пользователем ячейки в активную.
19. Присваивает активной ячейке указанное пользователем имя.
20. Устанавливает в активной ячейке произвольный денежный формат.
21. Записывает в активную ячейку символ авторского права ©.
22. Записывает в активную ячейку формулу вычисления площади круга, если радиус хранится в ячейке A1.
23. Записывает в активную ячейку формулу вычисления площади прямоугольника, если его высота хранится в ячейке A1, а ширина – в B1.
24. Записывает в активную ячейку символ Товарный знак ™.
25. Вставляет пустую строку перед активной ячейкой.
26. Вставляет пустой столбец перед активной ячейкой.
27. Удаляет строку в позиции активной ячейки.

28. Удаляет столбец в позиции активной ячейки.
29. Устанавливает в активной ячейке Финансовый формат.
30. Записывает в активную ячейку формулу вычисления площади прямоугольного треугольника, высота и ширина которого хранятся в ячейках A1 и B1, соответственно.

Контрольные вопросы

1. Что такое макрос? Для чего он используется?
2. Где Excel хранит записанные макросы?
3. Как в программе Excel выполняется записанный ранее макрос?
4. Как вы запускаете программу записи макросов?
5. Где вы сохраните макрос, если вы хотите, чтобы он был доступен для всех открытых книг Excel в любом сеансе работы?
6. Какие макросы вы увидите в списке диалогового окна Макрос после выполнения работы?
7. Назовите три основных окна VB-редактора. Опишите назначение каждого из них.
8. Что такое *модуль* VBA?
9. Что такое комментарии? Как их вставить в текст макроса?
10. Что такое *ключевое слово* VBA?
11. Какие необходимые элементы содержит процедура?
12. Что такое *тело процедуры* и где оно расположено?
13. Почему текст записанных макросов имеет отступы? Зачем делать отступы в текстах программ?
14. Для чего предназначена процедура MsgBox?

Лабораторная работа №2

«Линейный вычислительный процесс»

Изучим: типы данных, переменные, константы, операторы и выражения, стандартные функции VBA, обработка ошибок, отладка программы, окно отладки.

Познакомимся с типами данных

Тип данных – это термин, описывающий природу данных, которые VBA различает и умеет с ними обращаться. В таблице перечислены типы данных VBA и дано краткое описание типов и диапазон значений для каждого типа.

Таблица. Типы данных языка Visual Basic

Имя типа данных	*Размер в байтах*	*Описание и диапазон значений*
Byte	1 (8 бит)	Положительные числа от 0 до 255
Boolean	2 (16 бит)	Логические значения. Только значения True и False
Currency	8 (64 бит)	От -9223372036854775808 до 9223372036854775807
Date	8 (64 бит)	Хранит комбинацию даты и времени. Дата от 1 января 100 г. до 31 декабря 9999 г. Время от 00:00:00 до23:59:59
Double	8 (64 бит)	Отрицательные числа от - $1.79x10^{308}$ *по* - $4.9x10^{-324}$. Положительные числа от $4.9x10^{-324}$ до $1.79x10^{308}$
Integer	2 (16 бит)	Числа от -32768 до 32767
Long	4 (32 бит)	Числа от-2147483648 до 2147483647
Object	4 (32 бит)	Используется для доступа к любому объекту VBA. Хранит адрес объекта
Single	4 (32 бит)	Отрицательные числа от $-3.4x10^{38}$ до $-1.4x10^{45}$ Положительные числа от $1.4x10^{45}$ до $3.4x10^{38}$
String	1 байт на символ	Служит для хранения текста. Может содержать до 2 миллионов символов. Строки в программе заключаются в двойные кавычки.
Variant	16 байт + 1 байт	Служит для хранения данных любого типа

Вы можете преобразовывать большинство типов данных из одного вида в другой, кроме того, Visual Basic автоматически преобразует данные.

Данные типа Variant

Variant — это специальный тип данных, в котором можно хранить все типы данных, перечисленные в таблице, включая типы Object и Array. VBA использует тип данных Variant всегда, когда вы не указываете тип явно.

Данные типа Variant имеют свойства тех данных, которые фактически в них записаны. Например, если в данных типа Variant записана текстовая строка, они имеют свойства данных типа String, а если там записано число, то они имеют свойства числового типа, чаще всего типа Double, хотя это может быть и Integer, и Long, и Single, и Currency.

Данные типа Variant используют самое компактное из возможных представлений, например, если в данных типа Variant записано целое число, то тот эти данные будут трактоваться как Integer или как Long, в зависимости от размера этого числа. Число 15 в переменной типа Variant будет трактоваться как Integer, а число 1000000 — как Long.

VBA хранит числа с плавающей запятой в переменных типа Variant как Double и обращается с ними так, что вы не обязаны всегда заботиться о точном типе переменных в своих программах. Любая переменная, для которой вы не указали тип явно, становится переменной типа Variant.

Хотя использование переменных типа Variant удобно и освобождает вас от многих забот при написании процедур, такие переменные требуют больше памяти и большинство операций выполняется на них медленнее, чем с другими типами. Вообще говоря, вам следует избегать неоправданного использования переменных типа Variant, так как это может сделать ваши программы медленными и затруднить их чтение, кроме того, это может привести к появлению трудно обнаруживаемых ошибок.

Понятие о переменных

Переменная — это имя, которое вы, программист, дали участку компьютерной памяти для того, чтобы хранить в нем данные некоторого типа. Вы можете представить себе переменную как гнездо, в котором записано какое-то значение для дальнейшего использования. Имя переменной — это ярлык, по которому осуществляется доступ к участку памяти. Содержимое участка (значение переменной) может меняться, но имя остается неизменным. В переменной VBA могут храниться данные любого из типов, перечисленных в таблице.

В некотором смысле переменная подобна имени ячейки в листе Excel. Вы можете дать ячейке листа Excel имя и потом обращаться к этой ячейке по данному имени, не заботясь о том, какие номера столбца и строки имеет ячейка.

Имена переменных должны подчиняться следующим правилам.

- Имя переменной начинается с буквы.

- За буквой может следовать любая комбинация букв, цифр и символов подчеркивания.
- Имя переменной не должно содержать пробелов, точек и символов математических операций.
- Имя переменной не должно превышать 255 символов в длину.
- Имя переменной не должно совпадать ни с одним из ключевых слов (называемых также зарезервированными словами). Если такое совпадение будет иметь место, VBA выдаст сообщение о синтаксической ошибке.
- Имя переменной должно быть уникальным в процедуре. Если вы дадите двум переменным одинаковые имена или имя переменной совпадет с именем процедуры в том же самом модуле, при выполнении процедуры VBA выдаст сообщение об ошибке.

Примеры правильных имен переменных.

```
MyVar, PayDate, Percent, WholePart, Line12
```

А это примеры неправильных имен переменных.

```
New Iten        ‘Не разрешен символ пробела
5thDimention    ‘Начинается с цифры
Dim             'Совпадает с зарезервированным словом
Week/Day        'Содержит знак арифметической операции
```

Выбирая имена переменных, старайтесь делать их описательными. Гораздо лучше дать переменной имя Percent, чем X или Y. Например, если в переменной записывается температура в градусах по Цельсию, назовите переменную CelsiusTemp или DegreesC.

Создание переменных

Самый простой способ создать переменную — это просто упомянуть ее в выражении. VBA создает переменную и выделяет для нее память, как только обнаружит ее, обычно это происходит при записи в переменную некоторого значения.

Запись значения в переменную называется присваиванием значения переменной. Вы присваиваете значение переменной, используя оператор присваивания, который выглядит как знак равенства (=). Вот пример оператора присваивания.

```
MyVar=15
```

Этот оператор записывает число 15 в участок памяти, с которым связано имя переменной MyVar. Если это первое упоминание имени переменной, то VBA создает переменную, выделят для нее память и записывает в нее число 15.

Если такая переменная была создана ранее, то оператор присваивания просто записывает в участок памяти, на который указывает переменная, число 15, при этом все, что было записано в эту память ранее, безвозвратно стирается.

Такой способ создания переменной простым ее упоминанием называется неявным объявлением переменной. Упоминая переменную в выражении, вы

косвенно говорите программе VBA о том, что хотите создать такую переменную. Все переменные, созданные таким способом, получают тип Variant. Неявное создание переменных называют также созданием переменных "на лету" (on-the-fly declaration).

По некоторым причинам VBA предоставляет вам способ объявлять переменные явно, что избавит вас от возможных ошибок. Явное объявление переменных имеет много преимуществ:

- При явном объявлении переменных ваша программа работает заметно быстрее. VBA создает все переменные перед тем, как программа начнет работать, и на это не будет уходить время при работе программы.
- Явное объявление переменных помогает избежать многих ошибок, которые подстерегают вас при неявном объявлении.
- Явное объявление переменных облегчает чтение программы человеку. Видя в начале процедуры или модуля список всех переменных, человеку легче понять логику работы программы.
- При явном объявлении переменных вам не нужно следить за правильным использованием верхнего и нижнего регистров, так как за вас это будет делать VBA. Вместо того чтобы подгонять все вхождения имени переменной под последнее ее упоминание, VBA будет исправлять регистр имени в соответствии с тем написанием, которое было использовано при явном объявлении.

Для того, чтобы объявить переменную, используется инструкция Dim со следующим синтаксисом:

```
Dim имя
```

Имя — это любой идентификатор. Например:

```
Dim PcntProfit
```

Такая конструкция создает переменную с именем PcntProfit. (Ключевое слово Dim является сокращением от dimension — размерность.) Все переменные, которые создаются этим способом, имеют тип Variant. Имена нескольких переменных могут быть перечислены через запятую.

Объявлять переменную можно только один раз. Это может быть сделано в любом месте программы, но обязательно до первого упоминания переменной в процедуре.Однако хорошей практикой следует считать вынесение всех объявлений в один блок в начале программы.

В листинге приведена процедура, преобразованная так, что переменная HelloMsg объявляется явно. Процедура Hello выводит сообщение

Процедура Hello с явным объявлением переменных

```
1: Sub Hello ()
2: Dim HelloMsg   'текст сообщения для функции MsgBox
3: HelloMsg = "Hello, World!"
4: MsgBox HelloMsg, , "Приветствие"
5: End Sub
```

Указание типа переменной в инструкции Dim

Для того чтобы объявить тип переменной в инструкции Dim, добавьте за именем переменной ключевое слово As и укажите тип переменной. Синтаксис этой конструкции следующий.

Dim имя As тип

где имя в данном случае —идентификатор переменной, а тип — один из типов переменных, приведенных в таблице.

В следующих строках приведены примеры правильного объявления типа переменных.

Dim PcntProfit As Single

```
Dim Gross_Sales As Currency
Dim PayDay As Date
Dim Massage As String
Dim Counter As Integer
```

Объявляя переменную неявно, вы можете указать ее тип, добавляя в конце имени переменной специальный символ. Такие символы называются символами определения типа. В таблице перечислены символы определения типа и соответствующие им типы.

Символы определения типа

!	Single
@	Currency
#	Double
$	String
%	Integer
&	Long

Обратите внимание на то, что таких символов всего шесть. Не существует символов определения типа для таких типов, как Byte, Boolean, Date, Object или Array. Символы определения типа могут находиться только в конце имени переменной.

Понятие о константах.

Константа – это такая величина в программе, которая никогда не меняется. Константы могут быть поименованными и непоименованными. В общем случае следует использовать поименованную константу всегда, когда в программе встречается повторяющееся значение, или значение, которое трудно запомнить, или если само значение на момент написания программы неизвестно.

Имя константы подчиняется тем же правилам, что и имя переменной. Правила создания имен переменных мы рассмотрели выше в этой главе.

Как и в случае с переменной, константа должна быть объявлена до того, как вы первый раз к ней обратитесь. Но в отличие от переменной, константа

всегда объявляется явно, и служит для этого ключевое слово Const. Вот синтаксис объявления поименованной константы.

```
Const имя = значение
```

Значение может быть любого типа — числом, строкой или датой. Приведем несколько примеров правильного объявления констант.

```
Const        BillingPoint = 212
Const        Цена =8.25
Const        Приветствие ="Привет, коллеги!"
```

При желании можно объявить несколько констант в одной строке, разделяя их запятыми. Следующая строка полностью эквивалентна предыдущим трем.

```
Const BillingPoint = 212, Цена = 8.25, Приветствие = "Привет, коллеги!"
```

Вот пример правильного объявления констант.

```
Const BillingPoint = 212
Const DangerZone = BillingPoint+50
```

Объявлять переменную можно только один раз. Это может быть сделано в любом месте программы, но обязательно до первого упоминания переменной в процедуре. Однако хорошей практикой следует считать вынесение всех объявлений в один блок в начале программы.

Инструкция Dim появляется во второй строке листинга программы Hello. Когда VBA выполняет эту строку, он создает новую переменную и выделяет для нее память. (Переменная HelloMsg имеет тип Variant, поскольку специально ее тип не указан.) В этой же строке вы видите комментарии, объясняющие назначение этой переменной.

В строке 3 листинга присваивается значение переменной HelloMsg. В этой переменной содержится текст "Hello, World!". В следующей строке листинга переменная HelloMsg передается процедуре MsgBox, которая выводит на экран окно сообщения.

Когда вы создаете поименованную константу, VBA приписывает ей тип в соответствии с типом выражения, которое ей присваивается. Например, считается, что константа, содержащая строку, имеет тип String.

Но иногда вы можете указать тип константы явно. Это может понадобиться для повышения точности вычислений: константа, тип которой указан как Double, использует более широкий диапазон значений. Вы можете указать тип константы Integer, Long, Currency или другой для того, чтобы результат вычислений имел, в свою очередь, нужный тип.

Константа может иметь любой из типов, перечисленных для переменных, кроме Object и Array. Тип константы объявляется точно так же, как и тип переменной, просто объявление начинается со слова Const. Синтаксис объявления типа константы выглядит следующим образом.

```
Const имя As Тип = значение
```

Здесь имя — это идентификатор константы, тип — один из возможных VBA-типов, значение — это то значение, которое вы присваиваете константе. Вот пример правильного объявления константы.

```
Consr Pi As Double = 3.14
```

Здесь объявлена константа Pi, имеющая тип Double, и ей присвоено значение 3.14.

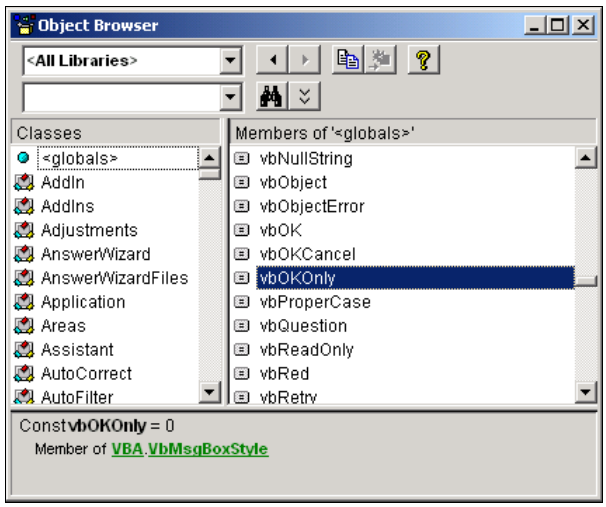
В VBA есть несколько констант, значение которых заранее определено для вас. Такие константы называют еще внутренними. Кроме внутренних констант VBA, существует несколько констант, определяемых приложением, из которого вызывается VBA. В частности, Excel 2000 определяет несколько констант для работы с книгами Excel, MS Word определяет константы для работы с документами и шаблонами, MS Access имеет несколько констант для работы с базами данных.

Все внутренние константы, определенные в VBA, имеют имя, начинающееся с букв vb. Например, такие константы, как vbOKOnly, vbOKCancel, vbAbortRetrylgnore определены в языке Visual Basic. В приложении Excel определены, например, такие константы: xlChart, xlCountrySetting, xlWorksheet. Как видите, их имена начинаются с букв xl.

Внутренние константы служат для упрощения работы с некоторыми встроенными про-. цедурами VBA, такими как уже знакомая вам MsgBox или InputBox, с которой вы познакомитесь позже. Процедура MsgBox имеет необязательный аргумент, указывающий количество кнопок в окне. Аргументы, соответствующие кнопкам, обычно задаются с помощью внутренних констант VBA.

Для того чтобы просмотреть полный список предопределенных констант как VBA, так и главного приложения, необходимо воспользоваться окном **Object Browser** (**Просмотр объектов**), которое активизируется:

- по команде **View ⇨ Object Browser**,
- по нажатию кнопки  **Object Browser** на панели инструментов **Standard**, или
- по нажатию клавиши **F2**.



Ввод данных пользователя в вашу процедуру

Вы уже научились пользоваться процедурой MsgBox для вывода на экран сообщений пользователю. Кроме того, вы познакомились с переменными и константами и теперь готовы к тому, чтобы научиться получать информацию от пользователя.

Получение данных, сохранение их в переменной, обработка этих данных или другой информации на основе этих данных — вот главные составляющие умения писать интерактивные программы. (Интерактивной называется программа, которая интенсивно общается с пользователем, выводя на экран информацию и получая информацию от пользователя.) Такие программы часто оказываются более эффективными, чем просто записанные макросы, так как пользователь может управлять их работой в зависимости от изменившихся условий.

Данные, полученные от пользователя, называются введенными данными, или просто вводом. Для получения данных от пользователя процедуры применяется функция InputBox. (Функция — это особый вид процедуры, которая возвращает некоторое значение.) Функция InputBox выводит на экран окно с текстом, предлагающим ввести некоторое значение, и полем ввода для этого значения. Кроме того, в окне функции InputBox присутствуют кнопки OK и Cancel.

Перед тем как вызывать функцию InputBox, вы должны сформировать текст приглашения. Кроме того, вы можете задать заголовок окна, которое выведет на экран InputBox.

Синтаксис вызова InputBox выглядит следующим образом.

```
Строка = InputBox (Приглашение [, Заголовок])
```

где строка — это любая переменная, в которой может храниться текстовая строка; может иметь тип String или Variant. Приглашение — поименованная или непоименованная константа или переменная, содержащая текст. Это обязательный аргумент, вы должны указывать его в вызове функции InputBox.

Заголовок — это необязательный второй аргумент функции InputBox. (Все необязательные элементы в описании синтаксиса берутся в квадратные скобки.) Заголовком также может быть поименованная или непоименованная константа или переменная, содержащая текст. Если вы опустите этот аргумент в вызове функции, в заголовке будет выведена строка Microsoft Excel. В листинге, приведенном ниже, вы видите модуль, содержащий единственную процедуру, и константу и переменную, объявленные на уровне модуля. Эта процедура вычисляет площадь круга. Эта процедура спрашивает у пользователя, для какого радиуса следует вычислять площадь.

Получение ввода пользователя с помощью функции InputBox

```
Dim CircleArea As Single   'сюда запишем вычисленную площадь
Const Pi As Single = 3.14   'приблизительное значение числа Pi..
Const BoxTitle = "Площадь круга"
```

```
   Sub Calc_CircleArea()
    Dim Radius As Single, Temp As String
6:      Temp = InputBox("Введите радиус круга", "BoxTitle")
7:      Radius = CSng(Temp)
8:      CircleArea = Pi * (Radius * Radius)
9:      MsgBox CircleArea, , "BoxTitle"
  End Sub
```

Процедура называется Culc_CircleArea.

В процедуре объявляются *константы*: Pi и BoxTitle, которая используется для вывода заголовков в окна, создаваемые этой процедурой, и *переменные* CircleArea, Radius, Temp.

Переменная CircleArea и константы Pi и BoxTitle объявляются за пределами процедуры, т.е. будут доступными всем процедурам данного модуля.

Обратите внимание на строку 6. В этой строке вызывается функция InputBox, которая выводит свой первый аргумент в качестве приглашения, поясняя пользователю, чего от него ждут. В данном случае выводится строка “Введите радиус круга”. Второй аргумент функции — это заголовок окна; в данном случае для заголовка окна используется переменная BoxTitle.

После вызова функции, возвращаемое ею значение присваивается переменной Temp; результат функции InputBox— это всегда строка, поэтому переменная Temp была объявлена с типом String. Если пользователь введет цифру и щелкнет ОК, в переменной Temp окажется строка, введенная в окне сообщения. Перед тем, как использовать результат в математических вычислениях, нужно преобразовать его в число. Это делается в следующей строке.

В строке 8 вычисляется площадь круга, а в строке 9 – выводится на экран вычисленное значение.

1. Запустите Excel и откройте свой файл.
2. Войдите в редактор VBA и отредактируйте записанную Вами процедуру Hello, введя явное описание переменных.
3. Запустите на выполнение процедуру Hello. Изменился ли результат выполнения процедуры?
4. Напишите программу, которая будет запрашивать значение переменной с экрана, считывать это значение и выводить его опять на экран вместе с именем переменной.

Понятие об операторах и выражениях

Выражение – это величина или группа величин, объединенных в единую сущность. Каждое выражение имеет определенное и единственное значение. В состав выражения могут входить: константы, переменные любого типа, операторы, массивы, элементы массивов, функции. Выражение может иметь

одно из особых значений — Empty (пусто) и Null. Для обозначения этих особых значений в языке VBA существуют соответствующие зарезервированные слова. Значение Empty соответствует неинициализированной переменной типа Variant или выражению, в которое входит такая переменная. Значение Null соответствует выражению, содержащему неверные данные, т.е. неопределенному выражению. Ниже приведено несколько примеров выражений.

Выражение	*Пояснение*
5	Значением является число 5
"5"	Значение — строка, состоящая из единственного символа 5
"сего"&"дня"	Значение — строка сегодня
C*(p/100)	Значение — число, полученное после умножения значения переменной C на значение, полученное после деления переменной P на константу 100
CStr(UOO)	Значение равно результату, возвращаемому функцией CStr. В данном случае это строка 1200 (функция CStr преобразует числа в строки)
MyValue <= 7	Логическое выражение. Его значение зависит от того, действительно ли значение переменной MyValue меньше или равно числу 7

Для объединения, сравнения или иного манипулирования значениями, входящими в выражение, используются операторы. Величины, входящие в выражение (это могут быть переменные или константы), называются операндами; большинство операторов требует наличия двух операндов. Например, в выражении 2+1 числа 2 и 1 - операнды, объединенные оператором сложения. В выражении может быть любое количество операторов.

Выражения могут входить в состав других выражений, сравниваться операторами сравнения или передаваться функциям и процедурам в качестве аргументов. Каждое выражение имеет единственное значение определенного типа. Выражения языка VBA в некотором смысле подобны предложениям человеческого языка. Каждая инструкция может содержать сколько угодно выражений. Рассмотрим типы выражений, которые встречаются в языке Visual Basic.

- Дата. Это любое выражение, которое имеет тип Date. Такие выражения могут содержать константы даты, переменные числового типа или типа Date, числовые константы и даты, возвращаемые функциями.

- Числовое выражение. Это выражение одного из следующих типов: Byte, Integer, Long, Single, Double или Currency. В числовое выражение могут входить числовые переменные и константы, функции, возвращающие числа, и знаки арифметических операций. Кроме того, эти выражения могут включать строки, если их можно преобразовать в число. (Строку можно преобразовать в число, если в нее входят только цифры; например, строку "36" можно преобразовать в число 36.)
- Строковые выражения. Это выражения имеющие тип String. Такое выражение может включать строковые константы и переменные, функции, возвращающие строки, и операторы конкатенации строк. В строковые выражения могут также входить числовые выражения, если их можно преобразовать в строки.
- Логические выражения. Это выражение, которое имеет одно из булевых значений, т.е. True или False. Такие выражения могут состоять из логических переменных и констант, функций, которые возвращают логическое значение, операторов сравнения и логических операторов.
- Объектное выражение. Это выражение, значением которого является ссылка на объект.

Не все типы данных совместимы друг с другом. Нельзя комбинировать несовместимые типы в одном выражении. Например, нельзя складывать текстовую строку "Привет!" с числом — такая операция не имеет смысла, и ее значение не определено.

В то же время некоторые типы являются совместимыми. Вы можете свободно комбинировать различные числовые типы между собой, VBA автоматически выполнит необходимые преобразования. Иногда VBA может автоматически преобразовывать и другие типы данных, добиваясь их совместимости, но не всегда.

Очень важно знать типы входящих в выражение значений, так как если в выражении встретятся несовместимые данные, то при выполнении программы VBA выдаст вам сообщение о несоответствии типов. Когда VBA встречает в выражении различные типы данных, он сначала пытается уладить конфликт, автоматически преобразуя данные к совместимым типам. Но если это не удается, вы получаете сообщение о несоответствии типов. То же самое произойдет, если вы попытаетесь присвоить переменной значение выражения, тип которого несовместим с типом переменной. Сообщение о несоответствии типов не выдается, если один из операндов имеет тип Variant. Как мы уже говорили, проще всего преобразование типов выполняется тогда, когда переменная имеет тип Variant, но не следует этим злоупотреблять.

Во многих процедурах вы можете видеть такую конструкцию.

```
Count = Count + 1
GrossTotal = GrossTotal + SubTotal
```

Это часто применяется для накопления значения какого-нибудь счетчика. Смысл написанного не сразу очевиден; на первый взгляд даже может показаться, что это противоречит логике. Но вы поймете написанное, если

вспомните, что при выполнении оператора присваивания сначала вычисляется выражение, стоящее справа от оператора, а потом полученное значение присваивается переменной, стоящей слева.

Оператор присваивания (=) служит для того, чтобы записать значение выражения в переменную или значение одной переменной в другую. Оператор присваивания имеет две синтаксические формы, которые обе законны и совершенно эквивалентны. В первом случае используется ключевое слово Let и синтаксис имеет следующий вид.

```
Let имя = выражение
```

где имя — идентификатор переменной, а выражение — любое допустимое в VBA выражение. Такая форма оператора присваивания пришла из старого языка Basic. Вот пример использования оператора присваивания.

```
Let X = Y        'Значение У записывается в переменную X
```

Вторая форма более общепринята и используется повсеместно. Синтаксис ее таков:

```
имя = выражение
```

где имя — идентификатор переменной, а выражение — любое допустимое в VBA выражение. Вот пример использования оператора присваивания в этой простой и общепринятой форме.

```
X = Y

MyVar = 12 - YourVar
```

В обеих формах оператора присваивания переменная слева от него получает значение выражения справа. При выполнении оператора присваивания сначала вычисляется выражение, стоящее справа от оператора, а потом полученное значение присваивается переменной, стоящей слева.

Во многих процедурах вы можете видеть такую конструкцию.

```
Count = Count + 1

GrossTotal = GrossTotal + SubTotal
```

Это часто применяется для накопления значения какого-нибудь счетчика. Смысл написанного не сразу очевиден; на первый взгляд даже может показаться, что это противоречит логике. Но вы поймете написанное, если вспомните, что при выполнении оператора присваивания сначала вычисляется выражение, стоящее справа от оператора, а потом полученное значение присваивается переменной, стоящей слева.

Арифметические операторы

VBA может выполнять все стандартные арифметические действия: сложение, вычитание, умножение и деление. Кроме того, есть оператор для возведения числа в степень и дополнительные операторы для целочисленного деления и для деления по модулю. В таблице собраны арифметические операторы VBA.

Оператор	*Синтаксис*	*Пояснение*
+	N1 + N2	Складывает N1 И N2
-	N1 -N2	Вычитает N2 из N1
- N1		Унарный минус (отрицательное число)
*	N1 * N2	Умножает N1 на N2
/	N1 / N2	Делит N1 на N2
\	N1 \N2	Целочисленное деление. Делит N1 на N2, отбрасывая дробную часть
Mod	Nl Mod N2	Деление по модулю. Делит N1 на N2, возвращая остаток от деления
^	N1 ^ N2	Возводит N1 в степень N2

5. Запишите программу, выполняющую вычисление площади круга с вводом радиуса с экрана и выводом результата на экран.
6. Запустите программу и выполните ее в пошаговом режиме.

Операторы сравнения

Операторы сравнения иногда называют операторами отношения. Обычно их применяют для задания критерия на принятие решения или для формирования условия выполнения некоторой последовательности операций. Результат оператора сравнения имеет тип Boolean и принимает только два значения — True и False. Сравнивать можно поименованные и непоименованные константы и переменные сходных типов.

Таблица. Операторы сравнения

Оператор	*Синтаксис*	*Описание*
=	E1 = E2	Равно. True, если E1 равно E2, False в противном случае
<	E1 < E2	Меньше. True, если E1 меньше E2, False в противном случае

<=	E1 <= E2	True, если E1 меньше или равно E2, False в противном случае
>	E1 > E2	True, если E1 больше E2, False в противном случае
>=	E1 >= E2	True, если E1 больше или равно E2, False в противном случае
<>	E1 <> E2	Не равно. True, если E1 не равно E2, False в противном случае
Is	El Is E2	Является. Оба операнда должны иметь тип Object. True, если E1 и E2 указывают на один и тот же объект, False в противном случае
Like	El Like E2	Подобно. Оба операнда должны иметь тип String. True, если El соответствует образцу, содержащемуся в E2, False в противном случае

Логические операторы

Обычно логические операторы применяются для того, чтобы, комбинируя условные выражения, создавать сложные условия для принятия решений в процедуре, или для формирования условий выполнения некоторой группы инструкций.

В качестве операнда в логическом операторе может использоваться любое значение типа Boolean или число, которое можно преобразовать к этому типу. VBA считает, что 0 соответствует значению False, а любое другое число — значению True.

Таблица Логические операторы

Оператор	*Синтаксис*	*Объяснение*
And	El And E2	Конъюнкция. Выражение истинно, если истинны оба его операнда
Or	El Or E2	Дизъюнкция. Выражение истинно, если истинен хотя бы один его операнд
Not	Not El	Отрицание. Выражение истинно, если E1 ложно, и наоборот

Таблица истинности оператора And.

Операнд1	*Операнд2*	*Значение выражения*
True	True	True
True	False	False
False	True	False
False	False	False

Оператор And используется для того, чтобы проверить, являются ли оба условия истиной одновременно. Например, рассмотрим оператор And в следующем выражении.

(Gross_Sales < 50000) And (Net_Profit < 10000)

Это выражение имеет значение True, если значение переменной Gross_Sales меньше 50000, а значение переменной Net_Profit меньше 10000.

Таблица истинности оператора Or.

Операнд1	*Операнд2*	*Значение выражения*
True	True	True
True	False	True
False	True	True
False	False	False

Оператор Or используется для того, чтобы проверить, является ли хотя бы одно из условий истиной. Например, рассмотрим оператор Or в следующем выражении.

(Grossjales < 50000) Or (Net_Profit < 10000)

Это выражение имеет значение True, если значение переменной Gross_Sales меньше 50000 или значение переменной Net_Profit меньше 10000.

Оператор Not называется логическим *отрицанием*. Он инвертирует любое значение, к которому применяется. Его значением будет True, если значение операнда — False, и наоборот.

Синтаксис оператора Not имеет следующий вид.

Not Операнд1

Операнд1 — это любое логическое выражение языка VBA.

Основной оператор конкатенации: "&".

Знак амперсанда имеет единственное применение в языке VBA — для слияния строк. Никаких других функций этот знак не выполняет. Синтаксис оператора конкатенации имеет следующий вид.

Операнд1 & Операнд2

Операнд1 и Операнд2 — это любые выражения строкового или числового типа, VBA String. Если один из операндов имеет значение Null или Empty, VBA трактует его как строку нулевой длины, т.е. строку, не содержащую символов.

Приоритет операций.

Приоритет операторов указывает порядок их обработки в сложном выражении. Если в выражении используется более одного оператора, то все они выполняются в строго определенном порядке, что обеспечивает однозначное трактование значения этого выражения.

В таблице, приведенной далее, представлены все операторы VBA в порядке уменьшения их приоритетов.

Приоритет	Операция	Описание
1	()	Круглые скобки
2	^	Возведение в степень
3	-	Унарный минус
4	*, /	Умножение и деление
5	\	Целочисленное деление
6	Mod	Деление по модулю
7	+, -	Сложение и вычитание
8	&	Конкатенация
9	>, <, >=, <=, =, <>, Is, Like	Операторы сравнения
10	Not	НЕ
11	And	И
12	Or	ИЛИ
13	Xor	Исключающее ИЛИ
14	Eqv	Эквивалентность
15	Imp	Импликация
16	=	Присваивание

7. Запишите программу согласно вашему варианту. Запустите программу на выполнение.
8. Отладьте программу. Откройте окно наблюдения переменных и выполните программу в пошаговом режиме. Составьте в тетради таблицу изменений значений переменных. Покажите результат преподавателю.

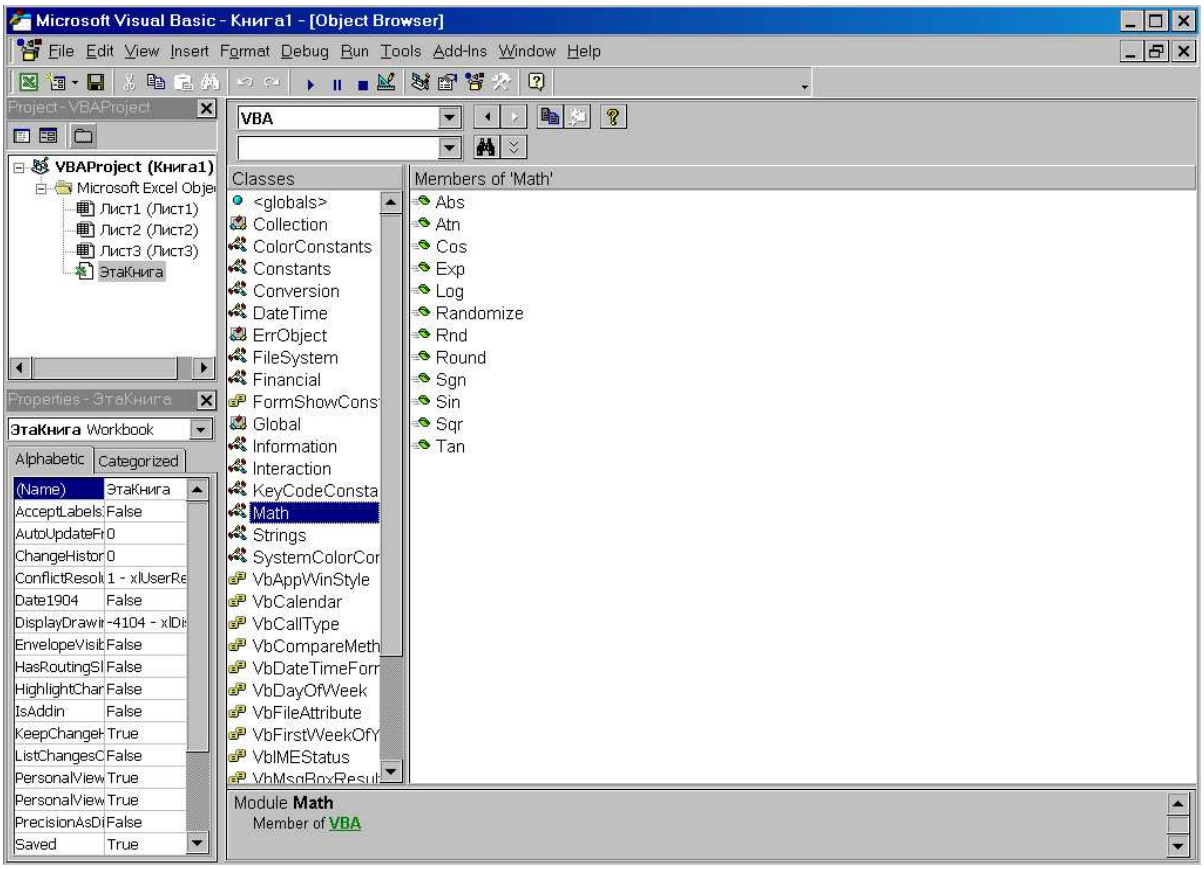
Индивидуальные Задания

№	***Задание***	№	***Задание***
1	$Y = a \cdot x + b;$	2	$Y = \cos(a) \cdot x + b;$
3	$Y = \sin(a) \cdot x + b / c;$	4	$Y = tg(a) \cdot x + b;$
5	$Y = a / x + b;$	6	$Y = (a + b) / x;$
7	$Y = tg(a) + b / c;$	8	$Y = \lvert a - b \rvert / x;$
9	$Y = \cos(a) + x / b;$	10	$Y = \lvert c - b \rvert / x;$
11	$Y = \cos(a) + b;$	12	$Y = \sin(a) + b;$
13	$Y = \sin(a) + b / c;$	14	$Y = \lvert c - b / a \rvert;$
15	$Y = \sin(a + b);$	16	$Y = a \cdot x + b;$
17	$Y = \cos(a + x) + b;$	18	$Y = \sin(a) / x + b / c;$
19	$Y = tg(a) \cdot x + b;$	20	$Y = a /(x + c);$
21	$Y = (a + b) / x;$	22	$Y = e^a + b / c;$
23	$Y = \lvert a - b \rvert / x;$	24	$Y = \sin(a + x / b);$
25	$Y = \lvert c - b / x \rvert;$	26	$Y = e^a + b;$
27	$Y = \sin(a + b / x);$	28	$Y = e^a + b / c;$
29	$Y = \sin \lvert c - b / a \rvert;$	30	$Y = \cos(a + b);$

Для того чтобы просмотреть полный список функций как VBA, так и главного приложения, необходимо воспользоваться окном **Object Browser** (**Просмотр объектов**), которое активизируется описанным выше способом. В окне **Classes** выберите Math, затем в окне **Members** выберите нужную вам функцию, просмотрите внизу синтаксис данной функции. Функция **Abs** вычисляет модуль аргумента, **Exp** – возводит *e* в степень, тригонометрические функции легко узнаваемы.

Открыть окно наблюдения значений переменных в редакторе VBA можно так: выделить переменную, значение которой вы хотите наблюдать,

затем выбрать ***Debug – Quick Watch***. Чтобы добавить для наблюдения еще одну переменную, нужно выделить ее и войти ***Debug – Add.***



Контрольные вопросы

1. Сколько числовых типов данных насчитывается в VBA?
2. Какая разница между числами типа Integer и Single? Между Integer и Long?
3. Что такое символы определения типа?
4. Что означает выражение определить переменную неявно? А что такое явно?
5. Каковы правила в языке VBA для имен переменных? Чем еще следует руководствоваться при выборе имен переменных?
6. Какие преимущества дает явное описание переменных?
7. Почему в процедурах лучше использовать поименованные константы?
8. Для чего служит функция InputBox?
9. Какие аргументы функции InputBox являются обязательными?
10. Какого типа значение возвращает функция InputBox?
11. Как можно воспользоваться результатом выражения? Должны ли вы использовать результат выражения?
12. Для каких целей используется знак равенства как оператор?

Упражнения

1. Исходя из здравого смысла, решите, какая из перечисленных ниже величин должна быть переменной, а какая — константой. Выберите для них имена и напишите определение типа (если можно, сделайте это и с помощью ключевых слов, и с помощью символов определения типа).

А Вычисленное количество столбцов в листе Excel
Б Объем продаж для подразделения компании
В Предполагаемое количество респондентов в опросе
Г Вычисленная площадь поверхности цилиндра
Д Коэффициент для перевода дюймов в сантиметры
Е Рентабельность операции, выраженная в процентах

2. Напишите вручную процедуру с названием EchoThis, в которой используется функция InputBox для ввода пользователем некоторой фразы, а потом эта фраза выводится на экран функцией MsgBox.

3. С помощью программы записи макросов создайте в книге Personal.xls макрос, который будет открывать существующую книгу Excel и выбирать в ней лист с именем Лист3. Остановите запись макроса. Отредактируйте записанный макрос так, чтобы он спрашивал у пользователя имя файла книги и затем открывал книгу с этим именем. (Подсказка. Сначала внимательно ознакомьтесь с текстом записанного макроса и поймите, в какой строке открывается файл книги. Поместите вызов функции InputBox непосредственно перед этой строкой. Присвойте введенное пользователем значение переменной, а затем подмените этой переменной непоименованную константу, созданную при записи макроса.)

4. Расставьте скобки в приведенных ниже выражениях так, чтобы их результат соответствовал указанному.

3*5-7	-6
4-7 + 26 /10	3.07
312 /47+16-2	5.114754

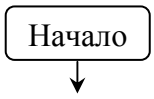
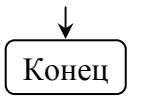
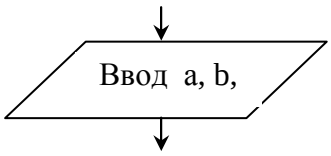
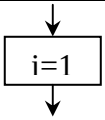
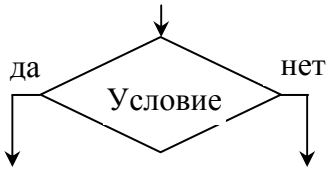
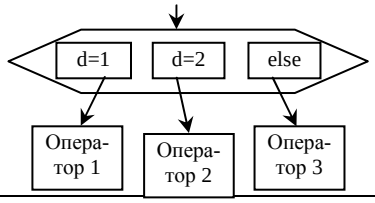
Лабораторная работа №3

«Разветвляющийся вычислительный процесс»

Алгоритмизация

Хорошую программу проще написать, если, предварительно проанализировав задачу, составить алгоритм ее решения, т. е. алгоритм программы. Алгоритм можно составить в виде словесного описания последовательности действий или представить в виде блок-схемы.

Блок-схема состоит из графических объектов различных типов, содержащих внутри себя текст, поясняющий, что именно выполняется в этом фрагменте. Графические объекты соединяются стрелками, показывающие последовательность обработки блоков. Перечень типов блоков и их описание приведен в следующей таблице.

№	Тип блока	Описание блока
1		*Признак начала* алгоритма.
2		*Признак конца* алгоритма.
3		*Блок ввода/вывода*, означает реализацию ввода данных с клавиатуры (или вывода данных на экран). В данном случае, это блок ввода, и эквивалентом его в VBA является оператор InputBox.
4		*Блок присвоения* переменным некоторых значений или выражений. В данном примере переменной i присваивается единица.
5		*Блок условия* применяется когда, в зависимости от истинности (или ложности) некоторого условия, выполняться будут те (или иные) операторы. Если *Условие* будет истинным, то выполняться будет левая ветвь, а правая выполняться не будет. Если *Условие* будет ложным, то выполняться будет правая ветвь, а левая выполняться не будет.
6	*Блок множественного ветвления* используется, когда в зависимости от значения некоторого ключа-идентификатора, вычислительный процесс разветвляется на больше чем две ветви. В VBA для этого используется оператор «Select Case». В данном примере образуется три ветви, но их может быть и больше. Вычислительный процесс идет только по одной ветви.	
Пример		«Else» является необязательным параметром и означает случай, когда ключ-идентификатор примет такое значение, которого нет среди перечисленных значений (для данного примера $d \neq 1$ и $d \neq 2$, например 3 или 0 или что-то другое).

Типы вычислительных процессов

1. Линейный вычислительный процесс

Если при запуске программы выполняются все ее операторы, начиная с первого и кончая последним, то это называется *линейным вычислительным процессом*. Например, вычисление площади круга, выполненное в предыдущей работе.

Освоению линейного вычислительного процесса посвящена вторая лабораторная работа.

2. Разветвляющийся вычислительный процесс.

Если в результате выполнения программы, процесс вычисления может пройти только по одному из нескольких (в общем случае многих) альтернативных путей, то такой вычислительный процесс называется разветвляющимся. В простейшем случае он реализуется оператором IF, который подразумевает две альтернативы: одну, если условие выполняется, и другую, если условие не выполняется. Например, если студент иногородний, то ему предоставляется общежитие, если нет, то общежитие не предоставляется.

Пример 1.

В некотором магазине предоставляется скидка в размере 10%, если сумма приобретенного товара превышает 1000 денежных единиц. Написать процедуру, которая будет подсчитывать сумму этой скидки.

Листинг процедуры **Discount1**, подсчитывающей сумму скидки, приведен далее.

```
1:Sub Discount1()
2:    Dim dCost As Double     ' Стоимость покупки
3:    Dim dDiscount As Double ' Скидка
4:
5:    dDiscount = 0#          ' Обнуление скидки
6:    dCost = InputBox("Введите стоимость покупки")
7:    If dCost > 1000 Then dDiscount = dCost * 0.1
8:    MsgBox "Скидка равна: " & dDiscount
9:End Sub
```

В ней:

Строки 2 и 3 – объявление двух переменных типа ***Double*** для хранения стоимости покупки и суммы скидки, соответственно.

Строка 5 – исходно никакая скидка не предоставляется.

Строка 6 – ввод стоимости приобретенного товара.

Строка 7 – подсчет суммы скидки, если стоимость покупки превышает 1000 денежных единиц.

Строка 8 – вывод результата.

Рассмотрим инструкцию ***If*** языка программирования VBA более формально, и более подробно. Синтаксис команды:

```
If <условие> Then <инструкции1> [Else <инструкции2>]
```

Работает инструкция ***If*** следующим образом:

1. В начале вычисляется значение `условия`, которое может быть ***True*** или ***False***.
2. Если `условие` имеет значение ***True***, то выполняется группа `инструкции`$_1$, причем несколько инструкций должны помещаться в одной строке, а группа `инструкции`$_2$ – пропускается.
3. Если же `условие` имеет значение ***False***, то, наоборот, выполняется группа `инструкции`$_2$, а группа `инструкции`$_1$ – пропускается.
4. Выполнение программы в любом случае продолжается со следующей за ***If*** инструкции.
5. Конструкции, заключенные в квадратные скобки, являются необязательными. То есть, может отсутствовать конструкция `Else` и `инструкции`$_2$, которые выполняются, если условие ложно.

В приведенном выше Примере 1 используется только одна инструкция `dDiscount = dCost * 0.1`, которая выполняется, если `условие` истинно, т.е. стоимость приобретенного товара превышает 1000 денежных единиц. Если же `условие` ложно, то никакие инструкции не выполняются.

Группы `инструкции`$_1$ и `инструкции`$_2$ могут состоять не из одной инструкции, как в предыдущем примере, а из нескольких, разделенных двоеточием (:). Например:

```
If A > 10 Then X = X + 1: Y = Y + X: Z = Z + Y
```

Приведенную выше процедуру **Discount1** можно написать и по-другому, например, задействовав ветвь ***Else*** инструкции ***If***. Листинг откорректированной таким образом процедуры, теперь уже названной **Discount2**, приведен далее.

```
10: Sub Discount2()
11:     Dim dCost As Double     ' Стоимость покупки
12:     Dim dDiscount As Double ' Скидка
13:
14:     dCost = InputBox("Введите стоимость покупки")
15:     If dCost > 1000 Then dDiscount = dCost * 0.1 _
16:                     Else dDiscount = 0#
17:     MsgBox "Скидка равна: " & dDiscount
18: End Sub
```

В ней значение скидки равное 0 присваивается переменной `dDiscount` не в начале процедуры априори, а только если сумма покупки не превышает 1000 денежных единиц. При такой записи процедуры длина инструкции ***If*** стала чересчур большой, и ее пришлось перенести на следующую строку (для переноса строки используется знак «_»). Если же в каждой ветви инструкции ***If*** будет задействовано по несколько инструкций, то вся инструкция ***If*** станет совсем нечитабельной. Для этого в VBA имеется блочная форма инструкции If, которая в общем виде выглядит следующим образом:

```
If <условие> Then
```

```
    <инструкции1>
[Else
    [<инструкции2>]]
End If
```

Работает она точно так же, как и приведенный ранее ее однострочный вариант.

Тогда, с учетом блочного синтаксиса инструкции ***If***, процедуру определения скидки можно записать следующим образом, назвав ее при этом уже **Discount3**, соответственно:

```
19:    Sub Discount3()
20:       Dim dCost As Double     ' Стоимость покупки
21:       Dim dDiscount As Double ' Скидка
22:
23:       dCost = InputBox("Введите стоимость покупки")
24:       If dCost > 1000 Then
25:          dDiscount = dCost * 0.1 _
26:       Else
27:          dDiscount = 0#
28:    End If
29:    MsgBox "Скидка равна: " & dDiscount
30: End Sub
```

Хотя процедура **Discount3** несколько длиннее своих предыдущих вариантов, зато она более структурирована и зрительно воспринимается гораздо лучше. *Однострочный вариант инструкции* ***If*** *следует использовать, если при выполнении (или невыполнении) некоторого условия необходимо исполнить одно простое действие* (как в Примере 1), *во всех же остальных случаях гораздо эффективнее применять ее блочный вариант.*

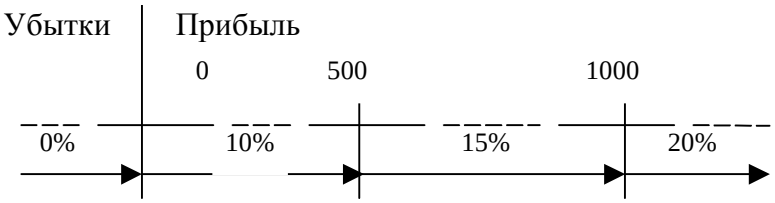
Язык программирования VBA допускает любую глубину вложенности инструкций `If` ***друг в друга.*** То есть в качестве `инструкции` в любой его ветви, в свою очередь, может выступать инструкция ***If***. Единственное и естественное возникающее при этом ограничение состоит в том, что *вложенная инструкция* ***If*** *должна целиком и полностью помещаться в соответствующей ветви внешней инструкции.*

1. Запустите Excel и откройте свой файл.
2. Войдите в редактор VBA и запишите процедуру, вычисляющую скидку на покупку в зависимости от стоимости товара по следующей схеме:
   a. Если стоимость товара<1000, скидка=0%;
   b. Если стоимость товара>=1000 и меньше 5000, скидка=10%;
   c. Если стоимость товара>=5000,скидка=20%.
3. Запустите на выполнение процедуру и выполните ее в пошаговом режиме, вводя значения стоимости товара из разных диапазонов. Обратите внимание, какие операторы выполняются в зависимости от вводимого значения.

4. Написать программу начисления налога на прибыль, используя вложенные операторы *If*, в соответствии со следующей таблицей:

Прибыль [денежная единица]	Ставка налога [%]
до 500	10
от 500 до 1000	15
1000 и более	20

Графически шкалу начисления налога, в зависимости от полученной прибыли, можно представить следующим образом:



Поскольку на убытки, или отрицательную прибыль, налог не начисляется, то в программе необходимо предусмотреть обработку такой ситуации. В ней так же необходимо предусмотреть две переменные типа **`Double`** для хранения: суммы прибыли и суммы начисляемого налога.

Вложенные инструкции **`If`** для проверки трех и более условий визуально становятся мало понятными. Поэтому в VBA была предусмотрена еще одна ее форма:

```
If <условие> Then
    [<инструкции1>]
[ElseIf <условиеn> Then
    [<инструкцииn>] ...
[Else
    [<инструкции2>]]
End If
```

В ней может быть сколько угодно ветвей **`ElseIf`** для реализации альтернативных вариантов. Единственное ограничение при этом – все они должны идти до последней ветви **`Else`**.

5. Составить блок-схему алгоритма в соответствии со своим вариантом. Ввод координат точки осуществить с экрана; в качестве заголовка окна ввода взять свою фамилию; определить, попадает ли точка в заданную область, и вывести на экран координаты точки и ответ о местоположении точки.

6. Написать и отладить программу разветвляющегося процесса с использованием оператора *If*.

Рассмотрим инструкцию ***Select Case*** языка программирования VBA, которая используется в том случае, если нужно выбрать более чем из двух ветвей выполнения процедуры. Синтаксис команды:

```
Select Case Выражение
      Case Логич.выражение1
            Инструкции1
      Case Логич.выражение2
            Инструкции2
      .
      .
      .
      Case Логич.выражениеN
            ИнструкцииN
      [Case Else
            ИнструкцииElse]
End Select
```

Работает инструкция ***Select Case*** следующим образом:

1. В начале вычисляется значение `Выражения`, и затем результат этого вычисления с каждым из выражений, хранящихся в каждом `Логич.выражении`.
2. Если значения `Выражения` удовлетворяет `Логич.выражению` для одного из разделов `Case`, `Инструкции` этого раздела выполняются.
3. Если `Выражение` соответствует более, чем одному разделу `Case`, выполняются инструкции только первого из них.
4. После выполнения инструкций выбранного раздела `Case` выполняется следующая за ключевыми словами `End Select` инструкция.
5. Если значения `Выражения` не удовлетворяет `Логич.выражению` ни для одного из разделов `Case`, выполняются инструкции `ИнструкцииElse`.

В отдельных разделах `Case` может стоять более, чем одна `Инструкция`, разделенные запятыми. `ИнструкцииN` имеют следующий синтаксис

```
Инструкция1, Инструкция2, ИнструкцияN
```

`Логич.выражения` могут быть некоторыми числовыми, строковыми или логическими выражениями, они могут быть представлены рядом значений с использованием оператора `To`:

```
Выражение1 To Выражение2.
```

Пример использования оператора `Select Case:`

```
Sub Temp()
    Dim temp
    Temp=Application.InputBox(_
                    Prompt:="Сколькоградусов?",_
                    Title:="Процедура Temp",_
                    Type:=1)
 Select Case temp
      Case Is>40
            MsgBox "Очень горячо!"
      Case 37 To 40
            MsgBox "Нормально!"
      Case 34 To 36
            MsgBox "Хорошо!"
      Case 30 To 33
            MsgBox "Прохладно!"
      Case Else
            MsgBox "Подогрейте!"
 End Select
End Sub
```

В языке программирования VBA существуют инструкции безусловного перехода, которые используется в том случае, если нужно перейти в какое-либо место процедуры без проверки условий; в основном эта инструкция применяется при обработке ошибок. Синтаксис команды:

```
GoTo Line,
```

где `Line` - метка или номер строки в той же процедуре, что и инструкция `GoTo.` Метка `Line` имеет следующий синтаксис:

```
name:
```

Здесь `name:`- это идентификатор VBA.

7. Написать и отладить программу разветвляющегося процесса с использованием оператора *Select Case* в соответствии со своим вариантом.
8. Показать результаты работы преподавателю.

ВАРИАНТЫ ЗАДАНИЙ

1. Попадает ли точка А в круг радиусом R=3 с центром в точке С(3, 4)?
2. Попадает ли точка А в круг радиусом R=5 с центром в точке С(6, 8)?
3. Попадает ли точка А в прямоугольник $2 \le x \le 4$ и $2 \le y \le 5$?
4. Попадает ли точка А в полосу $0 \le x \le 5$?
5. Попадает ли точка А в полосу $4 \le x \le 6$?
6. Превышает ли расстояние от точки А до точки В 5 единиц. Координаты точки В ввести с клавиатуры?
7. Лежит ли точка А на прямой $y = a + 4$?
8. Принадлежит ли точка А графику функции $y = 4x^2 - 5$?
9. Попадает ли точка А в прямоугольник $6 \le x \le 10$ и $7 \le y \le 9$?
10. Попадает ли точка А в один из кругов $R1 = 1$, С1(1, 1); $R2 = 2$, С2(5, 4)?
11. Попадает ли точка А в первый квадрант?
12. Попадает ли точка А во второй квадрант?
13. Попадает ли точка А в третий квадрант?
14. Попадает ли точка А в четвертый квадрант?
15. Попадает ли точка А в область положительных значений X?
16. Попадает ли точка А в область отрицательных значений X?
17. Попадает ли точка А в область неположительных значений оси y?
18. Попадает ли точка А в область неотрицательных значений оси y?
19. Попадает ли точка А в точку пересечения графиков функций $y = x + 1$ и $y = x^2 - 4 \cdot x + 2$?
20. Принадлежит ли точка А графику функции $y = 4x^2 + 8$?
21. Попадает ли точка А в область пересечения кругов С1: R1=3, центр в точке С1(0, 0) и С2: R2=5, С2(5, 0)?
22. Попадает ли точка А в прямоугольник $-1 \le x \le 5$ и $-2 \le y \le 3$?
23. Попадает ли точка А на окружность $R = 4,\ \ C(1,1)$?
24. Попадает ли точка А на окружность $R = 3,\ \ C(4,5)$?
25. Превышает ли расстояние от т. А до оси X 6 единиц?

26. Превышает ли расстояние от т. А до оси У 7 единиц?
27. Превышает ли расстояние от т. А до начала координат 5 единиц?
28. Попадает ли точка А на одну из окружностей: $R1 = 1,\ \ C1(1,2)$ и $R2 = 2,\ \ C2(2,3)$?
29. Превышает ли расстояние от т. А до прямой $x = 4$ шесть единиц?
30. Превышает ли расстояние от т. А до прямой $y = -5$ десять единиц?

Контрольные вопросы

1. Что такое условный и безусловный переход?
2. Какие вы знаете инструкции условного перехода?
3. Какие вы знаете инструкции безусловного перехода?
4. Когда и почему можно использовать инструкцию Select Case?
5. Сколько предложений Case может включать инструкция Select Case?
6. Как определить ветвь инструкции для выполнения, если в инструкции Select Case нет ни одного предложения Case? Куда в этом случае переходит выполнение внутри инструкции Select Case?
7. Что такое сложное условие? Как оно строится?
8. Сколько ветвей имеет оператор If Then?
9. Сколько ветвей имеет оператор Select Case?
10. Как перейти на нужную строку программы?

Лабораторная работа № 4

«Циклический вычислительный процесс»

Одним из недостатков записанных макросов является их неспособность выполниться несколько раз, пока это не будет выполнено вручную. В VBA имеется несколько мощных средств, позволяющих легко повторять операции и управлять их повторением.

Программные структуры, дающие возможность повторяться одной или нескольким инструкциям, называются циклами. Каждый раз, когда VBA завершает один полный цикл выполнения всех инструкций, находящихся внутри циклической структуры, называется итерацией цикла.

Некоторые циклические структуры созданы так, чтобы они выполнялись фиксированное число раз. Они называются определенными циклами. Другие выполняются неопределенное число раз, зависящее от некоторого условия. Ввиду того, что это число повторений неопределенно, они называются неопределенными циклами.

И в тех, и в других структурах имеются выражения, показывающие до каких пор, следует повторять цикл. Они называются определителями цикла. В фиксированной структуре такой определитель почти всегда является числовым выражением. В неопределенной структуре определитель обычно представляет собой логической выражение, которое описывает условие, при котором выполнение цикла продолжаются или останавливается. Логические выражения для циклических структур создаются и используются точно так же, как и в случае условных инструкций ветвления в VBA.

Принято два основных способа построения неопределенного цикла. В первом случае VBA будет проверять условия определителя до выполнения цикла. Если условие выполнения имеет значение False, то инструкции внутри цикла просто будет пропущены. Во втором случае можно создать такой цикл, чтобы условие в определителе проверялось после выполнения этих инструкции.

Можно также создавать неопределенный цикл, у которого в определителе нет никакого условия вовсе. Такого рода циклы выполняются вечно и называются бесконечными. Они никогда не заканчиваются; большинство таких циклов является результатом ошибки программиста.

Повторение фиксированное число раз: циклы For.

Простейшим из циклов является. В VBA имеется две различные структуры таких циклов: **For .. Next** и **For Each ... Next**. Обе эти структуры называются циклами **For**, потому что всегда выполняются определенное число раз.

Применение цикла For.. Next

Цикл For.. Next имеет такой синтаксис.

For *counter* = *start* **to** *end* **[Step** *StepSize***]**
Statements
Next [*counter***]**

Где **counter** представляет любую числовую переменную, соответствующую правилам VBA, обычно с типом данных Interger или Long. Выражение **start** является любым числовым выражением и определяет начальное значение переменной счетчика counter. Выражение **end** – также числовое выражение и определяет конечное значение той же переменной.

По умолчанию VBA каждый раз после завершения инструкций из цикла увеличивает значение переменной **counter** на 1. Другое значение увеличения можно задать с помощью необязательного ключевого слова **Step**. После него необходимо указать величину (шаг, на которую надо увеличить значение **counter**). В нашем синтаксисе **StepSize** представляет любое числовое выражение и как раз указывает эту самую величину.

Statements представляет одну или несколько инструкций VBA (или вообще ни одной из них). Они составляют тело цикла, и каждая из них выполняется всякий раз, когда выполняется сам цикл.

Ключевое слово **Next** сигнализирует VBA, что достигнут конец цикла, необязательная переменная **counter** после **Next** должна быть той же переменной, которая находится после ключевого слова **For** в начале циклической структуры. Ее надо писать после **Next**, чтобы легче читать исходный текст программы (особенно с вложенными циклами **For.. Next**).

При выполнении цикла **For.. Next** вначале переменной counter присваивается значение, представленное переменной **start**. Затем VBA выполняет все инструкции, представленные в **Statements**, пока не будет достигнуто ключевое слово Next. Ключевое слово Next сигнализируют VBA, что достигнут конец тела цикла. Затем значение переменной counter увеличивается на величину переменной **StepSize** (шаг), если есть необязательной ключевое слово **Step**. Если этого слова нет, то значение counter увеличивается на 1. Происходит переход на начало цикла, а затем текущее значение counter сравнивается со значением, представленным **end**. Если **counter** будет меньше или равен **end**, то цикл выполнится снова. Если **counter** больше, чем **end** , то выполнение продолжается с первой инструкции, которая находится после ключевого слова **Next**.

Счетчик в рассматриваемом типе циклов может быть как с возрастающим счетчиком, так и с убывающим. Для этого необходимо использовать ключевое слово Step, с отрицательным значением StepSize, в результате чего значение counter будет уменьшаться от большого значения к меньшему.

Пример цикла For.. Next с возрастающим счетчиком:

Эта процедура получает от пользователя два числа, в заданном ими диапазоне суммирует все целые числа, а затем выводит на экран полученное значение. Если, например, вы запустили эту программу и ввели значения 4 и 8, то будут просуммированы числа 4, 5, 6, 7 и 8, а на экране будет отображено число 30.

```
Sub Demo_ForNext()
Dim k As Integer
Dim uStart As String
Dim uEnd As String
Dim uSum As Long

uStart = InputBox("Введите натуральное число:")
uEnd = InputBox("Введите второе натуральное число")

uSum = 0

For k = uStart To uEnd
   uSum = uSum + k
Next k

MsgBox "Сумма натурального ряда от" & uStart & _
" до" & uEnd & " равна: " & uSum

End Sub
```

Введенные значения еще не обрабатывались, поэтому сумма должна быть нулевой

Цикл **For..Next**

Повторите назначение этого оператора

Пример цикла For.. Next с убывающим счетчиком:

Этот пример делает то же самое, но счетчик цикла работает с убыванием.

```
Sum Demo_ForNextDown()

Dim k As Integer
Dim uStart As Integer
Dim uEnd As Integer
Dim uSum As Long

uStart = InputBox("Введите натуральное число:")
uEnd = InputBox("Введите второе натуральное число")

uSum = 0
For k = uStart To uEnd Step -1
   uSum = uSum + k
Next k
```

Цикл **For..Next,** убывающий

```
MsgBox "Сумма натурального ряда от" & uStart & _
" до" & uEnd & " равна: " & uSum
End Sub
```

1. Введите оба примера, найдите отличия между ними.
2. Запустите их на выполнение, в качестве первого числа введите 3, а второго 15. Вычислите и запишите результат каждого примера.
3. Сделайте тоже самое, но числа введите наоборот, то есть сначала введите 15 а затем 3.
4. Поясните полученные результаты.

Повторение неопределенное число раз: циклы Do

В VBA имеется очень мощная инструкция для создания в процедурах и функциях циклических структур, повторяющихся неопределенное число раз. Все такие циклы создаются с помощью единственной инструкции — инструкции **Do**. Она обладает такими возможностями и такой гибкостью, что у нее фактически четыре конструкции, относящиеся к двум основным категориям.

Эти категории состоят, из циклов **Do**, которые проверяют условие определителя перед выполнением тела цикла, и, во-вторых, из тех циклов **Do**, которые проверяют это условие после выполнения тела цикла. Независимо от того, проверяется ил определитель до или после выполнения тела цикла, существует два способа контролировать, сколько раз выполняется неопределенный цикл.

- *Циклы, контролируемые счетчиками.* В этом случае тело цикла выполняется, пока значение некоторого счетчика находится ниже или выше указанной границы; в этом есть определенное сходство с циклов For .. Next, за исключением того, что на программисте лежит ответственность за инициализацию этого счетчика, а также за увеличение или уменьшение его значения. Такие циклы Do имеет смысл писать, когда счетчик не является регулярным, а также когда нельзя определить конечную границу, пока не началось выполнение самого цикла. Например, требуется пройти первые 16 строк листа, иногда увеличивая за один раз значение строки на 1, а иногда за один раз – на 2. В связи с тем, увеличение номера строка (то есть шаг счетчика) меняется, цикл For .. Next использовать нельзя; вместо него надо исследовать цикл Do.
- *Циклы, контролируемые событиями.* В этом случае значением определителя имеет значение True или False, в зависимости от некоторого события или действия, имевшего место во время выполнения цикла. Например, вы написали цикл, выполняемый неопределенное число раз, пока пользователь не ведет известное число в диалоговом окне. Ввод

такого числа как раз и является тем событием, которое завершит цикл. А вот другой пример: можно выполнять операции над ячейками листа, пока не будет достигнута пустая ячейка. Переход в пустую ячейку и будет событием, завершающим цикл.

Циклы с проверкой условия перед выполнением.

Чтобы условия определителя проверялись перед выполнением тела цикла, надо логическое выражение определителя просто поместить в начале инструкций, составляющих тело цикла. Это можно реализовать с помощью ключевых слов While и Until.

Создание циклов с помощью Do While.

Первой циклической структурой, где условие определителя проверяется перед выполнением цикла, является **Do While**, имеющего следующий синтаксис.

Do While *condition*
Statements
Loop

Где **condition** представляет логическое выражение для определителя цикла, **statements** — одну или большое количество инструкций VBA (или вообще ни одной из них), которые составляют тело цикла; все они выполняются каждый раз когда выполняется цикл. Ключевое слово **Loop**, находящееся после **statements**, сигнализирует что достигнут конец цикла, и указывает точку с которой выполняется переход в начало цикла, чтобы проверить условие определителя.

В инструкции **Do While** выражение **condition** находится в начале цикла, поэтому условие определителя проверяется перед выполнением самого цикла. Поскольку в этом случае используется ключевое слово **While**, цикл будет выполняться до тех пор, пока логическое выражение, представленное **condition**, будет иметь значение **True**.

Если логическое выражение, представленное condition, будет равно False при первом выполнении инструкции Do While, то цикл будет просто пропущен без всякого выполнения.

Пример цикла Do While

Эта процедура считает введенные пользователем нечетные числа, прекращая роботу тогда, когда таких чисел станет 10. Затем введенные числа выводятся на экран.

Обратите внимание на использование функции Mod.

```
Sub Count_OddNums()

  Const ocTitle = "Подсчет нечетных чисел"
  Dim OddCount As Integer ' счетчик нечетных чисел
  Dim OddStr As String ' строка для отображения нечетных чисел
  Dim Num ' переменная типа Variant для ввода

  OddStr = " " ' Строка для чисел пока что пуста
  OddCount = 0 ' Счетчик чисел пока что равен нулю

  Do While OddCount < 10

  Num = InputBox("", ocTitle)

          If (Num Mod 2) <> 0 Then
            OddCount = OddCount + 1
            OddStr = OddStr & Num & "  "
          End If
  Loop

  MsgBox "Вы ввели такие нечетные числа:" & Chr(13) _
            & OddStr, , ocTitle

  End Sub
```

Цикл **Do While…Loop**

5. Введите пример. Выполните его.
6. Как изменится процедура, если подсчитывать все числа кратные 3? а если нечетные? Ответ запишите в тетрадь.
7. Процедуру для ввода пяти чисел кратных 3 введите и проверьте.

Создание циклов с помощью Do Until.

В инструкции **Do** вариант **Do While** — это только один из тех способов ее создания, в которых условие определителя проверяется перед выполнением цикла. Вторым из этих способов является вариант **Do Until**, имеющий следующий синтаксис.

Do Until *condition*
statements
Loop

Где **condition** представляет логическое выражение для определителя цикла, а **statements** — инструкции VBA, которые составляют тело цикла. Ключевое слово **Loop**, находящееся после **statements**, сигнализирует, что достигнут конец

цикла, и указывает точку, с которой выполняется переход в начало цикла, чтобы проверить условие определителя.
В инструкции **Do until** выражение **condition** находится в начале цикла, поэтому условие определителя проверяется перед выполнением самого цикла. В связи с тем, что в этом случае используется ключевое слово **Until** (сто означает пока нет чего-либо), цикл будет выполняться до тех пор, пока логическое выражение, представленное condition, будет иметь значение **False**.
Если логическое выражение, представленное **condition**, будет равно **True** при первом выполнении инструкции **Do Until**, то цикл будет просто пропущен без всякого выполнения

Пример цикла Do Until

Эта шуточная процедура предлагает вводить вам дату в формате XX.XX.XX (например 18.03.03) до тех пор, пока вы не введете дату какого-нибудь воскресенья.

Обратите внимание на использование функции WeekDay.

```
Sub Stop_AtEvenNums()
Const evTitle = "В ожидании воскресенья..."
Dim EventFlag As Boolean  ' Признак выхода —Переменная для выхода из цикла
Dim Num ' Переменная для даты, описана как тип Variant

EventFlag = False ' Признак цикла пока равен ЛОЖЬ

Do Until EventFlag = True
Num = InputBox("Введите дату:", evTitle)

  If Weekday(Num, vbMonday) = 7 Then
     MsgBox "Вы ввели дату воскресенья!!!", , "Запомните эту дату! Ура!"
     EventFlag = True
  Else
     MsgBox " Жаль, но отдохнуть не удастся!", , "Не надо печалиться"
  End If
Loop
MsgBox "Хорошо повеселиться!!!", , "Мы рады за Вас!!!"
End Sub
```

Цикл
Do Until…Loop

8. Введите пример. Выполните его.
9. Добавьте в него еще одну проверку для субботы, с комментарием типа «Ура, это уже завтра!!!» и также выходом из цикла.

10. Найдите в справке описание функции WeekDay и законспектируйте в тетрадь ее аргументы.

Циклы с проверкой условия после выполнения.

Чтобы условия определителя проверялись после выполнения тела цикла логическое выражение определителя надо просто поместить в конце инструкции, составляющих тело цикла, а именно после ключевого слова Loop, которое сигнализирует о конце цикла.

Создание циклов с помощью Do…Loop While

Первой циклической структурой, где условие определителя проверяется после выполнения цикла, является **Do…Loop While**, имеющий следующий синтаксис.

Do
statements
Loop While *condition*

Где **condition** представляет логическое выражение для определителя цикла, **statements** — одну или большое количество инструкций VBA (или вообще ни одной из них), которые составляют тело цикла; все они выполняются каждый раз когда выполняется цикл. Ключевое слово **Loop**, находящееся после statements, сигнализирует что достигнут конец цикла, и указывает точку с которой выполняется переход в начало цикла, чтобы проверить условие определителя.

В инструкции **Do…Loop While** выражение **condition** находится в конце цикла, поэтому условие определителя проверяется после выполнения самого цикла.

Поскольку в этом случае используется ключевое слово **While**, цикл будет выполняться до тех пор, пока логическое выражение, представленное **condition**, будет иметь значение **True**.

Каким бы ни было значение логического выражения, представленного condition, этот цикл выполниться хотя бы один раз.

Создание циклов с помощью Do…Loop Until

Другой циклической структурой **Do**, где условие определителя проверяется после выполнения цикла, является **Do…Loop Until**, имеющего такой синтаксис.

Do
Statements
Loop Until *condition*

В этой инструкции выражение condition находится в конце цикла, поэтому условие определителя повторяется после выполнения самого цикла. Поскольку в этом случае используется ключевое слово **Until**, то цикл будет выполняться

до тех пор, пока логическое выражение, представленное condition, будет иметь значение **False**.

При выполнении цикла **Do…Loop Until** вначале выполняются инструкции, представленные в statements. Когда достигнуто ключевое слово **Loop**, проверяется логическое выражение, представленное **condition**; если его значение **False**, то происходит переход в начало структуры и снова выполняется тело цикла. Когда в конце цикла вновь достигнуто ключевое слово **Loop** и если значение **condition** сменилось на **True**, то выполнение продолжается с инструкций, находящихся после строи со словом **Loop**.

Каким бы ни было значение логического выражения, представленного condition, этот цикл выполниться хотя бы один раз.

Пример цикла Do…Loop Until

В процедуре приведенной ниже нужно угадать случайно сгенерированное число от 1 до 10. Цикл будет выводить окно, до тех пор, пока вы не угадаете «загаданное». В конце цикла, вы узнаете, сколько же попыток вам понадобилось.

Обратите внимание на генератор случайных чисел.

```
Sub Угадайка()
Dim EventFlag As Boolean
Dim g As Integer
EventFlag = False
Counter = 0  ’ счетчик ваших попыток
Randomize  ’ запуск генератора случайных чисел
c = Round(10 * Rnd + 1, 0)  ’ генерация случайного числа от 1 до 10
    Do
    g = InputBox("Угадайте число от 1 до 10")
        If g = c Then
        EventFlag = True
        MsgBox "Ты угадал с " & Counter & "  раза"
        End If
    Counter = Counter + 1
    Loop Until EventFlag
End Sub
```

11. Введите пример. Найдите в справке генератор случайных чисел и законспектируйте описание и примеры.
12. Подумайте, с помощью каких циклических конструкций этот пример записать можно, а с помощью каких нельзя. Ответ напишите в тетради и обоснуйте.

13. Запишите эту программу в тетради с помощью другой циклической конструкции по вашему выбору. Введите и выполните ее.
14. Выводите сообщения об удачливости игрока, в зависимости от числа попыток, например, при попытках больше 5 — «Что-то день сегодня не очень!», а меньше «Ну ты и везунчик!».

ВАРИАНТЫ ЗАДАНИЙ

1. Напишите программу, которая выводит все числа первой сотни, оканчивающиеся на 5.
2. Напишите программу, которая вводит число, а выводит таблицу умножения этого числа на все числа от 1 до 9.
3. Напишите программу, которая выводит таблицу степеней с 0-й по 9-ю числа 2.
4. Напишите программу, которая выводит таблицу значений квадратного корня на интервале [2;4] с шагом 0.1
5. Напишите программу, вывода таблицы квадратов первых 10 целых положительных чисел.
6. Напишите программу, вывода всех чисел, делящихся на 13 без остатка в интервале [1;100].
7. Напишите программу, вычисления среднего арифметического 10-и введённых чисел.
8. Напишите программу, которая 10 раз выводит Ваше имя и фамилию.
9. Напишите программу, вычисления суммы первых n целых положительных чисел. Количество суммируемых чисел n водится в начале работы программы.
10. Напишите программу, которая вычисляет сумму первых n членов ряда $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}$. Количество суммируемых членов ряда задаётся во время работы программы.
11. Напишите программу, которая выводит таблицу значений функции $y=-2.4x^2+5x-3$ в диапазоне[-2;2] с шагом 0.5.

12. Напишите программу, которая выводит таблицу значений функции $y=|x|$ в диапазоне[-4;4] с шагом 0.5.
13. Напишите программу, которая выводит таблицу значений функции $y=|x+2|$ в диапазоне[-4;4] с шагом 0.5.
14. Напишите программу, которая выводит таблицу значений функции $y=|x-2|+|x+1|$ в диапазоне[-4;4] с шагом 0.5.
15. Напишите программу, которая выводит таблицу стоимости яблок в диапазоне от 1 кг до 100 г с шагом 100 г. Стоимость 1 кг яблок вводится во время работы программы.
16. Напишите программу, подсчёта суммы и среднего арифметического всех целых положительных чисел из заданного диапазона. Диапазон задаётся во время работы программы.
17. Напишите программу, вычисления факториала $y=n!$ по формуле $n!=1*2*3*\cdots*(n-1)*n;\ 0!=1$. Значение n вводится во время работы программы.
18. Напишите программу вычисления $y=1!+2!+3!+\cdots+n!$. Ввод n осуществляется во время работы программы.
19. Напишите программу определения числа $e-$ основания натурального логарифма, с помощью ряда: $e=1+1/1!+1/2!+1/3!+\cdots+1/n!$ для всех значений n от 1 до m. Ввод m осуществляется во время работы программы.
20. Напишите программу вычисления $e^x=1+\frac{x}{1!}+\frac{x^2}{2!}+\cdots+\frac{x^n}{n!}+\cdots=1+\sum_{i=0}^{n}\frac{x^i}{i!}$ с заданной точностью $\varepsilon>0.0$, а также общее число слагаемых n. Каждое очередное слагаемое вычисляется через предыдущее по следующей рекуррентной формуле: $u_i=\frac{x}{i}u_{i-1}$, где $i=1,2,3$ и т.д. Ввод x и ε осуществляется во время работы программы.
21. Напишите программу, вычисления числа π с заданной точностью $\varepsilon>0.0$ при помощи следующего ряда: $\pi/4=1/1-1/3+1/5-1/7+1/9\cdots$, общий член которого вычисляется по формуле: $(-1^k)/(2k+1)$, где k – номер члена ряда.

Подсчитать и вывести при этом также общее число просуммированных членов ряда. Ввод ε осуществляется во время работы программы.

22. Напишите программу, определения варианта лабораторной работы по двум последним цифрам номера зачетной книжки. Вариант лабораторной работы вычисляется путем вычитания 30 из числа, образованного двумя последними цифрами номера зачетной книжки, до тех пор, пока оно больше 30. Две последние цифры номера зачётной книжки вводятся во время работы программы.
23. Напишите программу извлечения квадратного корня $y = \sqrt{x}$ методом Ньютона с заданной точностью $\varepsilon > 0.0$ по следующей рекуррентной формуле: $y_{n=1} = y_n + (x / y_n - y_n)/2$, где $n = 1,2,3$ и т.д. Ввод x и ε осуществляется во время работы программы.
24. Напишите программу определения значения величины: $y = \sum_{n=1}^{m} \frac{n}{n+1}$. Ввод m осуществляется во время работы программы.
25. Напишите программу определения значения величины: $y = \prod_{n=1}^{m} ((n+1)/n! + n)$. Ввод m осуществляется во время работы программы.
26. Напишите программу, которая последовательно вводит в переменную целого типа произвольные числа до тех пор, пока очередное введённое число не будет равно нулю, и определяет их сумму.
27. Напишите программу, которая последовательно вводит в переменную целого типа произвольные числа до тех пор, пока очередное введённое число не будет равно нулю, и определяет их среднее арифметическое.
28. Напишите программу, которая последовательно вводит n раз в переменную целого типа произвольные числа и определяет их среднее арифметическое.
29. Напишите программу печати таблицы перевода температуры по шкале Фаренгейта в температуру по шкале Цельсия в диапазоне от x до y с шагом z по формуле $C(F) = (5/9)(F - 32)$, где $C -$ температура по шкале

Цельсия, $F-$ температура по шкале Фаренгейта. Ввод x, y и z осуществляется во время работы программы.

30. Напишите программу печати таблицы перевода температуры по шкале Цельсия в температуру по шкале Фаренгейта в диапазоне от x до y с шагом z по формуле $F(C)=(9/5)(C+32)$, где $F-$ температура по шкале Фаренгейта, $C-$ температура по шкале Цельсия. Ввод x, y и z осуществляется во время работы программы.

Контрольные вопросы

1. Что такое циклы и для чего они предназначены?
2. Какие типы циклов вы знаете?
3. Что такое определитель цикла? В каких структурах он является числом?
4. Чем отличаются циклы предусловия и постусловия?
5. Запишите синтаксис всех циклов и охарактеризуйте его по такому плану: синтаксис; тип цикла; тип условия; характер определителя. Например:

Цикл	Тип цикла	Тип условия	Определитель
Do *Statements* **Loop Until** *condition*	неопределенный	послеусловие	условие

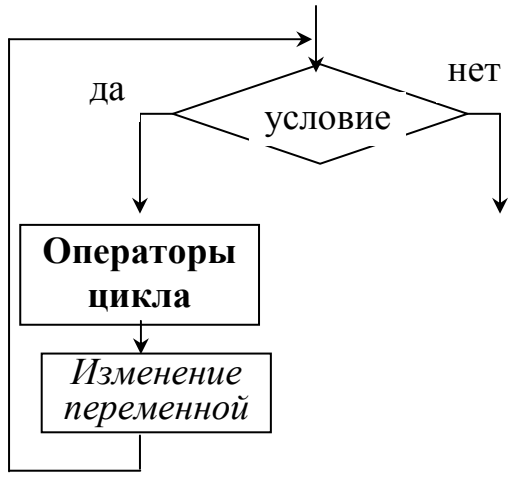
Лабораторная работа № 5

«Массивы»

Блок-схемы циклических структур

Суть *блока цикл* заключается в многократном повторении некоторых операторов (операторов цикла, т.е. операторов, находящихся в цикле). Блок цикл может быть изображен двумя способами: детальным и компактным.

Детальный способ

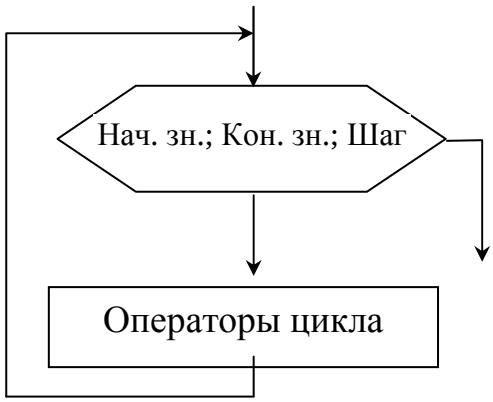


Если *Условие* истинно, то выполняются **Операторы цикла,** затем выполняется *Изменение переменной цикла*, после чего снова *проверяется Условие*. Эти действия выполняются много раз до тех пор, пока *Условие* не станет ложным. Если *Условие* ложно, то происходит выход из цикла.

Цикл используется для того, чтобы многократно повторять некоторые действия, которые и оформляются как **Операторы цикла**.

Следует отметить, что существует несколько модификаций этой схемы: «да» и «нет» могут меняться местами и «*Изменение переменной цикла*» может стоять перед проверкой «*Условия*». Эти модификации изменяют количество выполнений «Операторов цикла» и программист должен уметь составить цикл таким образом, чтобы «Операторы цикла» выполнились необходимое ему число раз.

Компактный



«Нач. зн.» означает начальное значение переменной цикла. «Кон. зн.» означает последнее значение переменной цикла, при котором **Операторы цикла** будут выполняться. «Шаг» определяет интервал между двумя соседними значениями переменной цикла.

1. Постройте блок-схемы для всех примеров процедур с циклическими процессами, созданными в предыдущей лабораторной работе

Понятие о массивах

Массив — это набор переменных одного типа данных с общим именем. Массивами обычно пользуются, чтобы представить списки или таблицы с информацией, где все элементы данные одного типа, то есть в списке находятся только числа Integer, данные одного из типов String, Currency и т.д. Массив дает возможность хранить и обрабатывать все свои элементы, используя одну переменную. Кроме того, что сокращается общее количество имен переменных, другим существенным преимуществом является то, что для обработки элементов в массиве можно использовать циклические структуры (в основном такие как For…Next или For…Each). С их помощью можно малым числом инструкций переработать горы данных. Если для тех же задач на каждый элемент данных задавать по переменной, то число инструкций возрастет до нескольких сотен.

Что такое одномерные массивы?

Наименее сложный массив — это список элементов данных: массив такого рода называется простым или одномерным. Одномерные массивы называются так потому, что у списка только одна мера — длина.

На рис. 1 показана схема одномерного массива.

10.

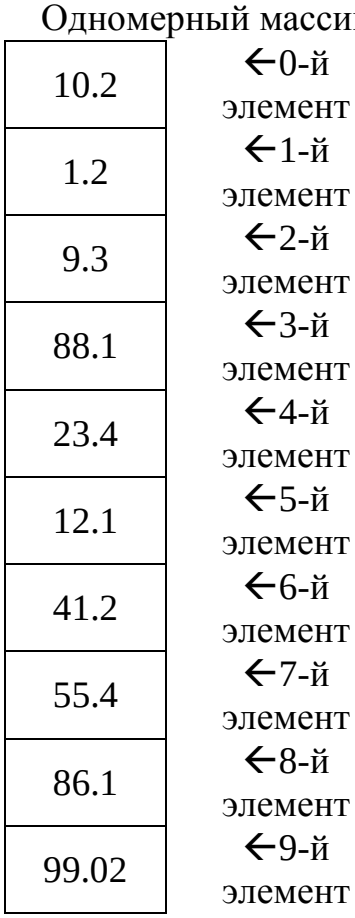


Рис. 1. Одномерный массив из 10 элементов типа Double с нумерацией от 0 до 9

Каждый «кирпичик» данных из массива называется его элементов. В массиве на рис. 1 содержится 10 элементов, в каждом из них хранится число

типа Double. Обратите внимание, что элементы массива, которых всего 10, пронумерованы от 0 до 9. (Этот способ нумерации распространен в программировании известен как нумерация с нижней границей — 0).

Чтобы получить доступ к отдельному элементу массива, указывается имя массива, за которым следует номер его элемента, чье содержимое надо взять или изменить (этот номер называется индексом). Индекс всегда надо заключать в скобки. Например, если массив на рис. 1 называется NumArray, то в следующей инструкции переменной AnyNum присваивается число 55,4.

```
AnyNum = NumArray(7)
```

В этой инструкции число 7 — это индекс массива; обратите внимание, что он заключен в скобки и никакими пробелами не отделен от имени массива. Ввиду того, что нумерация начинается с 0, эта инструкция в действительности ссылается на восьмой элемент указанного массива. Индексы используются и при внесении данных в отдельный элемент массива. В следующей инструкции число 12 запоминается в восьмом элементе массива (рис. 1):

```
NumArray(7) = 12
```

Когда VBA выполняет эту инструкцию, то помещает значение 12 в указанный элемент массива, затирая при этом находившееся в ней значение — опять же, как и при любом другом присвоении. Элемент массива можно использовать в любом выражении VBA точно так же, как и значение любой константы или переменной.

- **Индекс массива следует заключать в круглые скобки.**
- **В качестве индексов массивов требуется использовать целые числа.**

11. Не отделяйте пробелами имена массивов и индексы: в программе набирайте имя, скобки и индексы как одно слово

Статические и динамические массивы.

Обычно число элементов указывается при его описании. Описание сообщает VBA, какая величина у каждой из размерностей массива. После того как массив описан VBA выделяет под все его элементы нужное количество памяти. Для массива на рис.1 ее надо выделить столько, чтобы помещалось 10 чисел типа Double, для массива на рис. 2 нужно память под 20 таких чисел.

VBA резервируют память под элементы массива на все время существования его переменной. Такого рода массивы называются *статическими*, так как число элементов не меняется.

Выбор размера массивы может вызвать значительные трудности, когда неизвестно количество данных, которые будут в нем храниться, или когда это

число может сильно меняться. Если для этого иногда требуется 100 значений, а иногда — только 10, то место для 90 значений тратиться впустую (разница между наибольшим и наименьшим количеством элементов). Для такого рода массивов в VBA предусмотрен особый тип массивов, называемых *динамическими*. Своему названию они обязаны тому, что в них можно менять число элементов по ходу выполнения программы.

Описание массивов

Вам уже известна инструкция **Dim**, применяемая для описания переменных. Эта же инструкция используется для описания массивов. Фактически слово **Dim** является сокращением от английского Dimension — размерность.

При описании массива с помощью инструкции **Dim** используется следующий синтаксис.

Dim *VarName* ([*Subscripts*]) [**As** *Type*]

VarName представляет любое имя массива, соотвествующее стнадратным правилам VBA. **Subscripts** представляет собой размернгость (размерности) массива. Для одномерного массива в **Subscripts** описывается одна размерность; для двумерного — две размерности (выражения для каждой отделено запятой) и так далее, до того количества размерностей которое вам нужно.

Выражение **Subscripts** имеет такой синтаксис:

[lower To] upper[,[[lower To] upper]…

Переменная **Lower** указывает на нижнюю границу индекса массива, а upper — на верхнюю. Обратите внимание, что в выражении верхняя граница обязательна, а нижняя нет. Если будет указана только верхняя граница, то VBA будет нумеровать элементы массива от 0 до значения upper.

Если в выражении **Subscripts** будут находиться lower to, то это будет значительно облегчать понимание программ и находить в них ошибки, что сделает программы более понятными и более надежными. Использование lower To также дает возможность указать начальное значение индекса, которое отличается и от 0. Например, если есть необходимость создать массив с элементами, пронумерованными от 5 до 10 или от -5 до 0, от такая возможность существует.

Как и в стандартных переменных, тип данных можно описать и для массивов. Для этого в описание массива надо включить такое выражение как **As Type**. Тогда у каждого элемента массива будет указанный тип данных. Type представляет любой из возможных в VBA тип данных: Currency, Double, String и т.д. Если type будет пропущен, то у всех элементов массива будет тип Variant.

При инициализации элементы числовых массивов элементы получают значение 0, а в строковых — значение пустой строки.

Составить блок-схему алгоритма, написать, отладить и протестировать программу, реализующую следующее:

a) Сформировать случайным образом одномерные числовые массивы заданного размера и диапазона;
b) Выполнить заданные преобразования;
c) Вычислить некоторые величины;
d) Вывести массивы и значения вычисленных величин на экран.

ВАРИАНТЫ ЗАДАНИЙ

1. *a)* Одномерный массив из 10 элементов целого типа в диапазоне [-10, 20];
   b) Заменить в исходном массиве единицами элементы с четвертого по седьмой;
   c) Вычислить сумму оставшихся элементов массива;
   d) Вывести исходный, полученный массивы и значение суммы.
2. *a)* одномерный массив из 12 элементов целого типа в диапазоне [-1, 36];
   b) Образовать новый массив, перенеся в него четыре последних элемента исходного массива;
   c) Вычислить сумму элементов полученного массива;
   d) Вывести исходный, полученный массивы и значение суммы.
3. *a)* одномерный массив из 10 элементов целого типа в диапазоне [45, 100];
   b) Вставить два элемента по 1000, начиная с четвертого элемента исходного массива, сдвигая его элементы;
   c) Вычислить сумму элементов полученного массива;
   d) Вывести массив до, и после изменения и значение суммы.
4. *a)* Одномерный массив из 8 элементов целого типа в диапазоне [16, 40];
   b) Переставить четвертый элемент массива на последнее место, сдвигая элементы массива;
   c) Вычислить сумму элементов массива, начиная с пятого;
   d) Вывести массив до, и после изменения и значение суммы.
5. *a)* Два одномерных массива по 6 элементов целого типа в диапазоне [-10, 10];
   b) Переставить три первых элемента первого массива на три последних места второго массива;
   c) Вычислить суммы исходного и преобразованного массивов;
   d) Вывести два исходных массива, измененный массив и значение суммы.
6. *a)* Два одномерных массива по 6 элементов целого типа в диапазоне [1, 40];
   b) Переставить на второе место во втором массиве, сдвигая его элементы, второй элемент первого массива;

c) Найти минимальный элемент преобразованного второго массива;
d) Вывести исходный, полученный массивы и минимальный элемент.

7. *a)* Два одномерных массива по 6 элементов целого типа в диапазоне [10, 20];
   b) Переставить на пятое место в первом массиве, сдвигая его элементы, последний элемент второго массива;
   c) Найти максимальный элемент преобразованного второго массива;
   d) Вывести исходный, полученный массивы и максимальный элемент.
8. *a)* Одномерный массив из 10 элементов целого типа в диапазоне [-14, 80];
   b) Сдвинуть элементы исходного массива на семь значений вправо, освобождая семь мест, и заполнить их нулями;
   c) Найти максимальный элемент преобразованного второго массива;
   d) Вывести исходный, полученный массивы и максимальный элемент.
9. *a)* Два одномерных массива по 5 элементов целого типа в диапазоне [-100, 100];
   b) Переставить на третье место в первом массиве, сдвигая его элементы, четвертый элемент второго массива;
   c) Вычислить сумму элементов полученного массива;
   d) Вывести исходный, полученный массивы и значение суммы.
10. *a)* Два одномерных массива по 5 элементов целого типа в диапазоне [-4, 18];
    b) Переставить четвертый элемент первого массива на четвертое место во втором массиве, первый массив сжать;
    c) Найти произведение элементов первого массива;
    d) Вывести два исходных, два полученных массива и значение произведения.
11. *a)* Одномерный массив из 10 элементов целого типа в диапазоне [50, 200];
    b) Удалить из исходного массива элементы со второго по седьмой, сдвигая его элементы;
    c) Вычислить сумму элементов полученного массива;
    d) Вывести исходный, полученный массивы и значение суммы.
12. *a)* Два одномерных массива по 5 элементов целого типа в диапазоне [-80, -30];
    b) Поменять местами третьи элементы этих массивов;
    c) Вычислить сумму элементов первого массива до и после преобразования;
    d) Вывести исходный, полученный массивы и значение сумм.
13. *a)* Одномерный массив из 6 элементов целого типа в диапазоне [-10, 20];
    b) Удалить из исходного массива второй элемент и сжать массив (значение удаленного элемента напечатать);
    c) Найти минимальный элемент преобразованного массива;
    d) Вывести исходный, полученный массивы и значение минимального элемента.
14. *a)* Два одномерных массива по 5 элементов целого типа в диапазоне [16, 80];

b) переставить четыре последних элемента первого массива на четыре первых элемента во втором массиве;
c) Найти минимальный элемент преобразованного второго массива;
d) Вывести исходный, полученный массивы и значение минимального элемента.

15. *a)* Одномерный массив из 8 элементов целого типа в диапазоне [-5, 20];
b) Заменить в исходном массиве два первых элемента единицами;
c) Вычислить сумму элементов массива, начиная с третьего;
d) Вывести исходный, полученный массивы и значение суммы.
16. *a)* Одномерный массив из 5 элементов целого типа в диапазоне [-4, 100];
b) Дополнить исходный массив, начиная с третьего номера, пятью двойками, сдвигая элементы массива;
c) Найти максимальный элемент преобразованного массива;
d) Вывести исходный, полученный массивы и значение максимального элемента.
17. *a)* Два одномерных массива по 5 элементов целого типа в диапазоне [-10, 30];
b) Дополнить второй массив, начиная со второго номера, тремя последними элементами первого массива;
c) Найти максимальный элемент преобразованного второго массива;
d) Вывести два исходных, два полученных массивы и значение максимального элемента.
18. *a)* Одномерный массив из 10 элементов целого типа в диапазоне [40, 120];
b) Удалить из исходного массива три первых элемента, сдвигая элементы массива, напечатать их значения;
c) Найти произведение элементов преобразованного массива;
d) Вывести исходный, полученный массивы и значение произведения.
19. *a)* Одномерный массив из 9 элементов целого типа в диапазоне [-30, -1];
b) Переставить шестой элемент исходного массива на первое место, сдвигая его элементы, вывести его значение;
c) Вычислить произведение элементов преобразованного массива;
d) Вывести исходный, полученный массивы и значение произведения.
20. *a)* Одномерный массив из 12 элементов целого типа в диапазоне [-50, 50];
b) Удалить из исходного массива три первых элемента и четыре последних, вывести значения удаляемых элементов;
c) Найти максимальный элемент преобразованного массива;
d) Вывести исходный, полученный массивы и значение суммы.
21. *a)* Одномерный массив из 10 элементов целого типа в диапазоне [40, 70];
b) Заменить пять последних элементов исходного массива числом 100;
c) Найти максимальный элемент преобразованного массива и его порядковый номер в массиве;
d) Вывести исходный, полученный массивы, значение максимального элемента и его порядковый номер.
22. *a)* Одномерный массив из 10 элементов целого типа в диапазоне [-80, -50];

b) Сдвинуть элементы исходного массива на 5 значений к началу, заполнив пустые места единицами;
c) Вычислить сумму элементов нового массива;
d) Вывести исходный, полученный массивы и значение суммы.

23. *a)* Одномерный массив из 10 элементов целого типа в диапазоне [0, 40];
b) Дополнить исходный массив тремя единицами, начиная с пятого элемента, не сдвигая элементы исходного массива;
c) Найти минимальный элемент преобразованного массива;
d) Вывести исходный, полученный массивы и значение минимального элемента.
24. *a)* Одномерный массив из 10 элементов целого типа в диапазоне [-17, 60];
b) В конец исходного массива добавить шесть девяток;
c) Найти максимальный элемент преобразованного массива;
d) Вывести исходный, полученный массивы и значение максимального элемента.
25. *a)* Два одномерных массива (по 5 элементов в каждом) из целых чисел в диапазоне [-30, 18];
b) Заменить два последних элемента первого массива четвертым и шестым элементами первого массива;
c) Найти минимальный элемент до и после преобразований;
d) Вывести исходный, полученный массивы и значения минимальных элементов.
26. *a)* Одномерный массив из 7 элементов целого типа в диапазоне [-5, 100];
b) Инвертировать исходный массив (изменить знаки элементов на противоположные);
c) Вычислить сумму элементов полученного массива;
d) Вывести исходный, полученный массивы и значение суммы.
27. *a)* Одномерный массив из 10 элементов целого типа в диапазоне [4, 46];
b) Образовать новый массив, перенеся в него элементы исходного массива с 4 по 9 элементы включительно;
c) Вычислить сумму элементов полученного массива;
d) Вывести исходный, полученный массивы и значение суммы.
28. *a)* Одномерный массив из 9 элементов целого типа в диапазоне [-18, 42];
b) Присвоить первому элементу массива значение 0, остальные элементы сдвинуть;
c) Вычислить сумму элементов полученного массива;
d) Вывести исходный, полученный массивы и значение суммы.
29. *a)* Одномерный массив из 5 элементов целого типа в диапазоне [-32, -5];
b) Поставить на первые три места в исходном массиве -1, сдвигая его элементы;
c) Найти минимальный элемент преобразованного массива;
d) Вывести исходный, полученный массивы и значение минимального элемента.
30. *a)* Одномерный массив из 8 элементов целого типа в диапазоне [4, 36];
b) Поменять местами четвертый и седьмой элементы исходного массива;

c) Найти максимальный элемент преобразованного массива;

d) Вывести исходный, полученный массивы, значения переставленных элементов и максимальный элемент.

ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ

31. *a)* Одномерный массив из 10 элементов целого типа в диапазоне [-20, 80];

b) Второй массив составить из первого путем перенесения в него нечетных элементов исходного массива. Вывести полученный массив на экран. Затем удалить из этого массива третий по номеру элемент, сдвигая его элементы;

c) Вычислить сумму элементов исходного массива и найти минимальный элемент преобразованного массива;

d) Вывести исходный, два полученных массива и значения суммы и минимального элемента.

1) Текст программы с пояснениями.

```
Sub ArrayOne()
' Массив First статический с 10 элементами от 0 до 9
Dim First(9) As Integer
' Массив Second динамический
Dim Second() As Integer
'Выполняем пункт a)
'Заполнение исходного массива случайными числами
Randomize 'Запуск счетчика случайных чисел
For i = 0 To 9
' Присвоить элементу массива случайное число от 1 до 100 со сдвигом на -20
First(i) = Int((100) * Rnd - 21)
Next i
'Заполнить строку элементами массива
'перемежая их пробелами и вывести на экран исходный массив
For i = 0 To 9
FirstStr = FirstStr & First(i) & "  "
Next i
MsgBox FirstStr, , "Исходный массив"

'Выполняем пункт b)
' Подсчитать количество нечетных чисел для определения размерности
'массива Second
For j = 0 To 9
   If (First(j) Mod 2) <> 0 Then
   k = k + 1
   End If
Next j
'Вывести на экран количесвто нечетных элементов
```

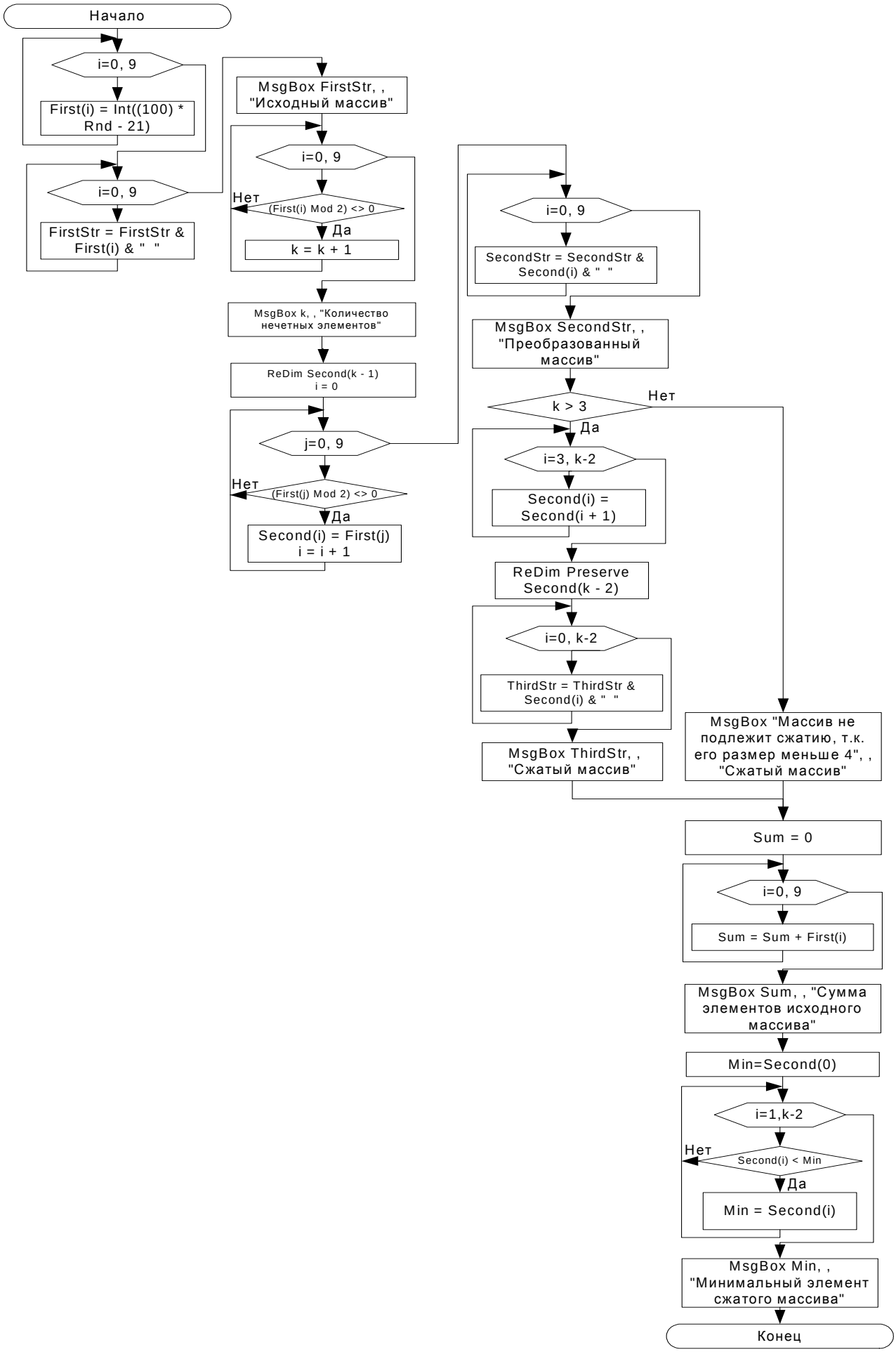
```
MsgBox k, , "Количество нечетных элементов"
'Определить размерность массива Second из k элементов от 0 до k-1
ReDim Second(k - 1)
  i = 0 'счетчик массива Second обнулить
  For j = 0 To 9
   If (First(j) Mod 2) <> 0 Then
      'Записать в массив Second нечетное число из массива First
    Second(i) = First(j)
  'счетчик массива Second увеличивается только для нечетных чисел массива First
    i = i + 1
    End If
Next j
'Заполнить строку элементами мас сива перемежая их пробелами и вывести на
'экран второй массив с нечетными элементами
 For i = 0 To k - 1
    SecondStr = SecondStr & Second(i) & "  "
Next i
MsgBox SecondStr, , "Преобразованный массивы"
'Если величина массива Second больше 3 то  выбросить из него элемент с номером 3
If k > 3 Then
'Начиная с элемента с номером три присваивать ему следующий элемент этого же массива
'Чтобы не выйти за пределы масссива заканчиваем'на предпоследнем элементе.
    For i = 3 To k - 2
      Second(i) = Second(i + 1)
    Next i
    ' Так как количество элементов в массива Second на единицу уменьшилось
    ' изменяем его размерность на 1. Ключевое слово Preserve не позволит
    ' обнулить уже существующие элементы массива
    ReDim Preserve Second(k - 2)
    'Сформировать строку для вывода сжатого массива на экран и вывести его на экран
        For i = 0 To k - 2
    ThirdStr = ThirdStr & Second(i) & "  "
    Next i
    MsgBox ThirdStr, , "Сжатый массив"

Else
' Если в массиве Second менее 4 элементов, он сжатию не подлежит
MsgBox "Массив не подлежит сжатию, т.к. его размер меньше 4", , "Сжатый массив"
```

```
End If
'Выполняем пункт c)
'Подсчитываем сумму элементов исходного массива
Sum = 0
For i = 0 To 9
'Добавляем к сумме каждый элемент исходного массива
Sum = Sum + First(i)
Next i
MsgBox Sum, , "Сумма элементов исходного массива"
'Находим минимальный элемент
'Принимаем за минимальный элемент нулевой элемент массива
Min = Second(0)
For i = 1 To k - 2
'Начиная с первого элемента массива проверяем, если он меньше
'минмального, то записываем его в минимальное значение
If Second(i) < Min Then
Min = Second(i)
End If
Next i
MsgBox Min, , "Минимальный элемент сжатого массива"
'Вывести на экран все полученное в програмне все вычисленное в программе уже один раз ‘выведено на экран, поэтому следующую команду можно не выполнять, но выводя на экран ‘все одновременно легче проверить правильность работы всей программы
MsgBox "Исходный массив: " & FirstStr & Chr(13) & _
    "Преобразованный массив: " & SecondStr & Chr(13) & _
    "Сжатый массив: " & ThirdStr & Chr(13) & _
    "Сумма элементов исходного массива: " & Sum & Chr(13) & _
    "Минимальный элемент сжатого массива: " & Min _
    , , "Итоги работы программы"
End Sub
```

2) Таблица результатов тестирования.

Исходный X	30 56 -5 27 6 -15 0 43 78 -10
Преобразованный Y	-5 27 -15 43
Сжатый Y	-5 27 -15
s=	210
min=	-15

3) Создание блок-схемы алгоритма.


Начало
i=0, 9
First(i) = Int((100) * Rnd - 21)
i=0, 9
FirstStr = FirstStr & First(i) & " "
MsgBox FirstStr, , "Исходный массив"
i=0, 9
Нет
(First(i) Mod 2) <> 0
Да
k = k + 1
MsgBox k, , "Количество нечетных элементов"
ReDim Second(k - 1)
i = 0
j=0, 9
Нет
(First(j) Mod 2) <> 0
Да
Second(i) = First(j)
i = i + 1
i=0, 9
SecondStr = SecondStr & Second(i) & " "
MsgBox SecondStr, , "Преобразованный массив"
k > 3
Нет
Да
i=3, k-2
Second(i) = Second(i + 1)
ReDim Preserve Second(k - 2)
i=0, k-2
ThirdStr = ThirdStr & Second(i) & " "
MsgBox ThirdStr, , "Сжатый массив"
MsgBox "Массив не подлежит сжатию, т.к. его размер меньше 4", , "Сжатый массив"
Sum = 0
i=0, 9
Sum = Sum + First(i)
MsgBox Sum, , "Сумма элементов исходного массива"
Min=Second(0)
i=1,k-2
Нет
Second(i) < Min
Да
Min = Second(i)
MsgBox Min, , "Минимальный элемент сжатого массива"
Конец

Лабораторная работа № 6

«Двумерные числовые массивы»

Вложенные циклы

Циклы можно размещать внутри других циклов точно так же, как это делается с инструкциями If…Then. Размещенные таким образом циклы обычно называются вложенными. Вложенные циклические структуры могут быть любого типа (смешанные циклы For и Do) и могут располагаться на любом уровне.

У каждого из таких циклов должна быть уникальная переменная счетчика, если она используется в данной циклической конструкции.

Многомерные массивы

Одномерные массивы хорошо работают с простыми списками данных. Однако зачастую в программах приходится представлять информацию, которая находится в ячейках таблицы, имеющих строки и столбцы — как если бы она находилась в ячейках Excel. Для этого необходимо использовать многомерный массив.

На рис. 1 представлена наиболее распространенная форма многомерного массива — двумерного массива.

Двумерный массив

Столбец 0	Столбец 1	
12.4	10.2	Строка 0
2.5	1.2	Строка 1
16.7	9.3	Строка 2
8.18	88.1	Строка 3
0.125	23.4	Строка 4
21.45	12.1	Строка 5
17.8	41.2	Строка 6
62.37	55.4	Строка 7
7818.01	86.1	Строка 8
37.2	99.02	Строка 9

Рис. 2. Двумерный числовой массив размерности 2 на 10 с нумерацией от 0 до 1 и от 0 до 9

Многомерные массивы называются так из-за того, что у них более одной размерности: длина (число строк), ширина (число столбцов). На рис. 1. у двумерного массива два столбца (номер 0 и номер 1) и 10 строк (пронумерованные от 0 до 9), а элементов всего 20. Как и в случае с одномерными массивами, для доступа к элемента многомерных массивов используется их индексация, то есть для указания отдельного элемента

применяют номер его столбца и номер строки. Индексация двумерного массива во многом напоминает указание ячейки листа Excel; первая размерность массива соответствует столбцам, а вторая — строкам.

Если у массива на рис. 1 имя NumTable, то следующая инструкция присваивает значение 10,2 (из первого ряда второго столбца массива) переменной AnyNum:

```
AnyNum = NumTable(1,0)
```

Аналогичным образом следующая инструкция присваивает значение 2,5 элементу из второго ряда первого столбца массива:

```
NumTable(1,0)=2,5
```

Обратите внимание, что в обоих выражения индексы массива заключены в круглые скобки, и координаты столбца и строки отделены запятыми.

Можно создавать массивы более чем с двумя размерностями; фактически VBA дает возможность задавать вплоть до 60 размерностей. Но работа с массивами 3, 4 и более размерностями является кропотливой и утомительной. Но к счастью, в программировании используются в основном одно- и двумерные массивы.

- Индексы многомерного массива необходимо отделять запятыми.
- Не отделяйте пробелами имена массивов и индексы: в программе набирайте имя, скобки и индексы как одно слово.

Составить блок-схему алгоритма и программу, реализующую следующее:

1. Сформировать случайный двумерный массив заданного размера и диапазона;
2. Выполнить заданные преобразования;
3. Вычислить некоторые величины;
4. Вывести массивы и значения вычисленных величин на экран.

ВАРИАНТЫ ЗАДАНИЙ

1. *a)* Двумерный массив размерностью 5 на 8 из целых элементов в диапазоне [-30, 30];
   b) Найти максимальный элемент второй строки;
   c) Вычислить сумму элементов третьего столбца;
   d) Вывести исходную матрицу, элементы третьего столбца и значения максимального элемента и вычисленную сумму.
2. *a)* Матрицу размерностью 6 на 5 из случайных элементов действительного типа, выведенных с точностью до десятых, в диапазоне [-80, 30];
   b) Транспонировать исходную матрицу;
   c) Найти минимальный элемент на главной диагонали;
   d) Вывести исходную и транспонированную матрицы и минимальный элемент.
3. *a)* Матрицу размерностью 9 на 6 из случайных целых элементов в диапазоне [-50, 50];
   b) Найти сумму элементов шестого столбца и седьмой строки;
   c) Найти минимальную из двух сумм;
   d) Вывести на экран исходную матрицу, обе суммы и минимальную из них.
4. *a)* Матрицу размерностью 4 на 4 из случайных элементов действительного типа, выведенных с точностью до сотых, в диапазоне [-20, 20];
   b) Сделать исходную матрицу симметричной относительно главной диагонали;
   c) Найти произведение элементов главной диагонали;
   d) Вывести на экран исходную, симметрическую матрицы и произведение элементов.
5. *a)* Матрицу размерностью 8 на 7 элементов действительного типа, выведенных с точностью до десятых, в диапазоне [-60, 40];
   b) Найти элементы меньшие 0 и заменить их нулями;
   c) Сосчитать количество замененных элементов;
   d) Вывести на экран исходную и преобразованную матрицы, и количество замененных элементов.
6. *a)* Матрицу размерностью 6 на 8 из элементов целого типа в диапазоне [-80, 80];
   b) Из исходной матрицы образовать подматрицу размером 4 на 5, начиная со второй строки и второго столбца исходной;
   c) Найти сумму элементов подматрицы;
   d) Вывести на экран исходную матрицу, подматрицу и сумму элементов.
7. *a)* Массив размерностью 5 на 8 из элементов целого типа в диапазоне [-50, 40];
   b) Сформировать из исходной матрицы вектор размерностью 1 на 5;

c) Нечетные элементы вектора находятся как минимальные элементы соответствующей строки, а четные элементы - как максимальные элементы соответствующих строк;

d) Вывести на экран исходную матрицу и полученный вектор.

8. *a)* Матрицу размерностью 8 на 9 из элементов целого типа в диапазоне [-90, 80];
   b) Сформировать из исходной матрицы вектор размерностью 9 на 1;
   c) Элементами вектора являются количества положительных элементов соответствующих столбцов исходной матрицы;
   d) Вывести на экран исходную матрицу и полученный вектор.
9. *a)* Матрицу размерностью 5 на 6 из элементов действительного типа с точностью до десятых в диапазоне [-40, 40];
   b) Найти количество положительных элементов исходной матрицы;
   c) Элементы на главной диагонали заменить единицами;
   d) Вывести на экран исходную, преобразованную матрицы и количество положительных элементов.
10. *a)* Матрицу размерностью 10 на 8 из элементов целого типа в диапазоне [-16, 43];
    b) Из исходной матрицы образовать вектор 1 на 4;
    c) Первый элемент вектора - сумма элементов второй строки матрицы, второй - разность элементов четвертой строки, третий элемент - произведение элементов пятого столбца, четвертый - частное элементов седьмого столбца;
    d) Вывести на экран исходную матрицу и полученный вектор.
11. *a)* Матрицу размерностью 8 на 6 из элементов действительного типа с точностью до сотых в диапазоне [4, 16];
    b) Определить количество элементов меньших 10;
    c) Максимальный элемент матрицы заменить этим числом;
    d) Вывести на экран исходную и преобразованную матрицы .
12. *a)* Матрицу размерностью 6 на 6 из элементов целого типа в диапазоне [-20, 20];
    b) Заменить местами второй столбец и третью строку;
    c) Для замены использовать дополнительный вектор;
    d) Вывести на экран исходную, преобразованную матрицы и дополнительный вектор.
13. *a)* Матрицу размерностью 5 на 8 из элементов целого типа в диапазоне [-50, 60];
    b) Изменить знаки элементов матрицы на противоположные;
    c) Сосчитать количество нулевых элементов;
    d) Вывести на экран исходную, преобразованную матрицы и количество нулей.
14. *a)* Матрицу размерностью 6 на 5 из элементов целого типа в диапазоне [-100, 100];

b) Подсчитать количество отрицательных элементов;
c) Заменить третью и четвертую строки исходной матрицы количеством отрицательных элементов;
d) Вывести на экран исходную и преобразованную матрицы.

15. *a)* Матрицу размерностью 7 на 7 из элементов действительного типа (с точностью до сотых) в диапазоне [0, 50];
b) Сделать зеркальное отображение относительно элементов третьего столбца;
c) Подсчитать количество положительных элементов;
d) Вывести на экран исходную, преобразованную матрицы, вектор обмена и количество положительных элементов.
16. *a)* Матрицу размерностью 8 на 8 из элементов целого типа в диапазоне [-50, 50];
b) Подсчитать количество элементов больших нуля и количество элементов меньших нуля;
c) Максимальный элемент матрицы заменить количеством элементов больших нуля, а минимальный - количеством элементов меньших нуля;
d) Вывести на экран исходную, преобразованную матрицы.
17. *a)* Матрицу размерностью 9 на 4 из элементов действительного типа (с точностью до десятых) в диапазоне [-80, 40];
b) Все элементы матрицы увеличить вдвое;
c) Найти сумму элементов главной диагонали;
d) Вывести на экран исходную, преобразованную матрицы и сумму элементов главной диагонали.
18. *a)* Матрицу размерностью 8 на 10 из элементов целого типа в диапазоне [-40, 40];
b) Из исходной матрицы получить подматрицу размерностью 8 на 5 элементов, начиная с элемента X[1, 1];
c) Найти максимальный элемент подматрицы;
d) Вывести на экран исходную матрицу, подматрицу и максимальный элемент.
19. *a)* Матрицу размерностью 5 на 6 из элементов целого типа в диапазоне [-40, 50];
b) Поменять местами элементы пятой строки и пятого столбца;
c) Найти сумму элементов вектора обмена;
d) Вывести на экран исходную, преобразованную матрицы и сумму элементов вектора обмена.
20. *a)* Матрицу размерностью 4 на 10 из элементов действительного типа (с точностью до десятых) в диапазоне [-20, 20];
b) Найти минимальный элемент матрицы;
c) Умножить элементы второго и четвертого столбцов на минимальный элемент;

d) Вывести на экран исходную, преобразованную матрицы и минимальный элемент.

21. *a)* Матрицу размерностью 6 на 9 из элементов целого типа в диапазоне [-60, 40];
    b) Четным элементам матрицы заменить знак на противоположный;
    c) Найти произведение четных элементов матрицы;
    d) Вывести на экран исходную, преобразованную матрицы и произведение.
22. *a)* Матрицу размерностью 9 на 6 из элементов целого типа в диапазоне [-30, 80];
    b) Образовать вторую матрицу размерностью 2 на 2 и заполнить ее нулями;
    c) Заменит элементы исходной матрицы, начиная с пятой строки и четвертого столбца, элементами второй матрицы;
    d) Вывести на экран исходные матрицы и преобразованную матрицу.
23. *a)* Матрицу размерностью 7 на 8 из элементов целого типа в диапазоне [-50, 50];
    b) Образовать вторую матрицу размерностью 7 на 7 и заполнить ее случайными числами в диапазоне от 10 до 20;
    c) Вычесть из первой матрицы вторую поэлементно;
    d) Вывести на экран исходную и преобразованную матрицы.
24. *a)* Матрицу размерностью 5 на 6 из элементов действительного типа с точностью до десятых в диапазоне [-10, 10];
    b) Подсчитать количество положительных элементов;
    c) Положительные элементы заменить единицами;
    d) Вывести на экран исходную, преобразованную матрицы и количество положительных элементов.
25. *a)* Матрицу размерностью 10 на 8 из элементов целого типа в диапазона [-50, 60];
    b) Найти сумму элементов, для которых X[i, j]>0;
    c) Элементы, для которых X[i, j]>0, уменьшить в 10 раз;
    d) Вывести на экран исходную, преобразованную матрицы и сумму элементов.
26. *a)* Матрицу размерностью 5 на 5 из элементов целого типа в диапазоне [-50, 60];
    b) Преобразовать исходную матрицу в симметрическую, оставив без изменения элементы, стоящие над диагональю;
    c) Найти разность элементов, не изменивших своего местоположения;
    d) Вывести на экран исходную, преобразованную матрицы и разность.
27. *a)* Матрицу размерностью 8 на 8 из элементов целого типа в диапазоне [-60, 50];
    b) Преобразовать исходную матрицу в симметрическую, оставив без изменения элементы, стоящие под главной диагональю;
    c) Заменить элементы главной диагонали нулями;

d) Вывести на экран исходную, преобразованную матрицы (до и после изменения главной диагонали).

28. *a)* Матрицу размерностью 5 на 8 из элементов целого типа в диапазоне [-50, 60];
    b) Транспонировать исходную матрицу;
    c) Подсчитать количество положительных и отрицательных элементов матрицы;
    d) Вывести на экран исходную, преобразованную матрицы и количество положительных и отрицательных элементов отдельно.
29. *a)* Матрицу размерностью 7 на 8 из элементов целого типа в диапазоне [10, 60];
    b) Найти минимальный и максимальный элементы исходной матрицы;
    c) Преобразовать исходную матрицу таким образом, чтобы максимальный элемент стоял на месте минимального, а минимальный на месте максимального;
    d) Вывести на экран исходную, преобразованную матрицы.
30. *a)* Матрицу размерностью 10 на 7 из элементов целого типа в диапазоне [-60, 60];
    b) Подсчитать количество положительных элементов в исходной матрице;
    c) Всем положительным элементам изменить знак на противоположный;
    d) Вывести на экран исходную, преобразованную матрицы и количество положительных элементов в исходной матрице.

ПРИМЕР ВЫПОЛНЕНИЯ ЗАДАНИЯ

31. *a)* Матрицу размерностью 5 на 7 из случайных элементов действительного типа, заданных с точностью до сотых в диапазоне [-30, 40];
    b) Найти минимальные элементы в нулевой, второй и четвертой строках и образовать из них одномерный массив;
    c) Найти сумму минимальных элементов и преобразовать исходный массив таким образом, чтобы элементы второго столбца полученной матрицы были уменьшены на полученную сумму;
    d) Вывести на экран исходную, полученную матрицы, матрицу минимальных элементов и их сумму.

1) Текст программы с пояснениями.

```
{Работа с двумерными числовыми массивами.}
Sub ArraySecond()

' Исходный Массив First статический двумерный (матрица) размерностью 5 на 7
' с элементами занумерованными от 0 до 4 и от 0 до 6
Dim First(4, 6) As Single
```

```
' Массив Help статический вспомогательный массив  с нумерацией ' элементов от 0 до 2 для
' трех минимальных значений из строк с номерами 0, 2, 4
Dim Help(2) As Single

' Преобразованный массив Second статический двумерный (матрица) размерностью 5 на 7
' с элементами занумерованными от 0 до 4  и от 0 до 6
Dim Second(4, 6) As Single

'Выполняем пункт a)Заполнение исходного массива случайными числами
'
Randomize 'Запуск счетчика случайных чисел
For i = 0 To 4
 For j = 0 To 6
' Присвоить элементу массива случайное число от 1 до 70 со сдвигом на -31
First(i, j) = Round((70) * Rnd - 31, 2)
Next j
Next i

'Заполнить строку элементами массива перемежая их пробелами  и началом новой строки
'по перемене индекса i как счетчика строки вывести на экран исходный массив

For i = 0 To 4
 For j = 0 To 6
FirstStr = FirstStr & First(i, j) & "   "
Next j
FirstStr = FirstStr & Chr(13)
Next i
MsgBox FirstStr, , "Исходный массив"

'
'Выполняем пункт b), то есть находим минимальное значение 'для строк 0, 2, 4 и записываем эти значения в массив Help
'
' Для строки 0
Help(0) = First(0, 0)
For j = 0 To 6
If Help(0) > First(0, j) Then Help(0) = First(0, j)
Next j

' Для строки 2
Help(1) = First(2, 0)
For j = 0 To 6
```

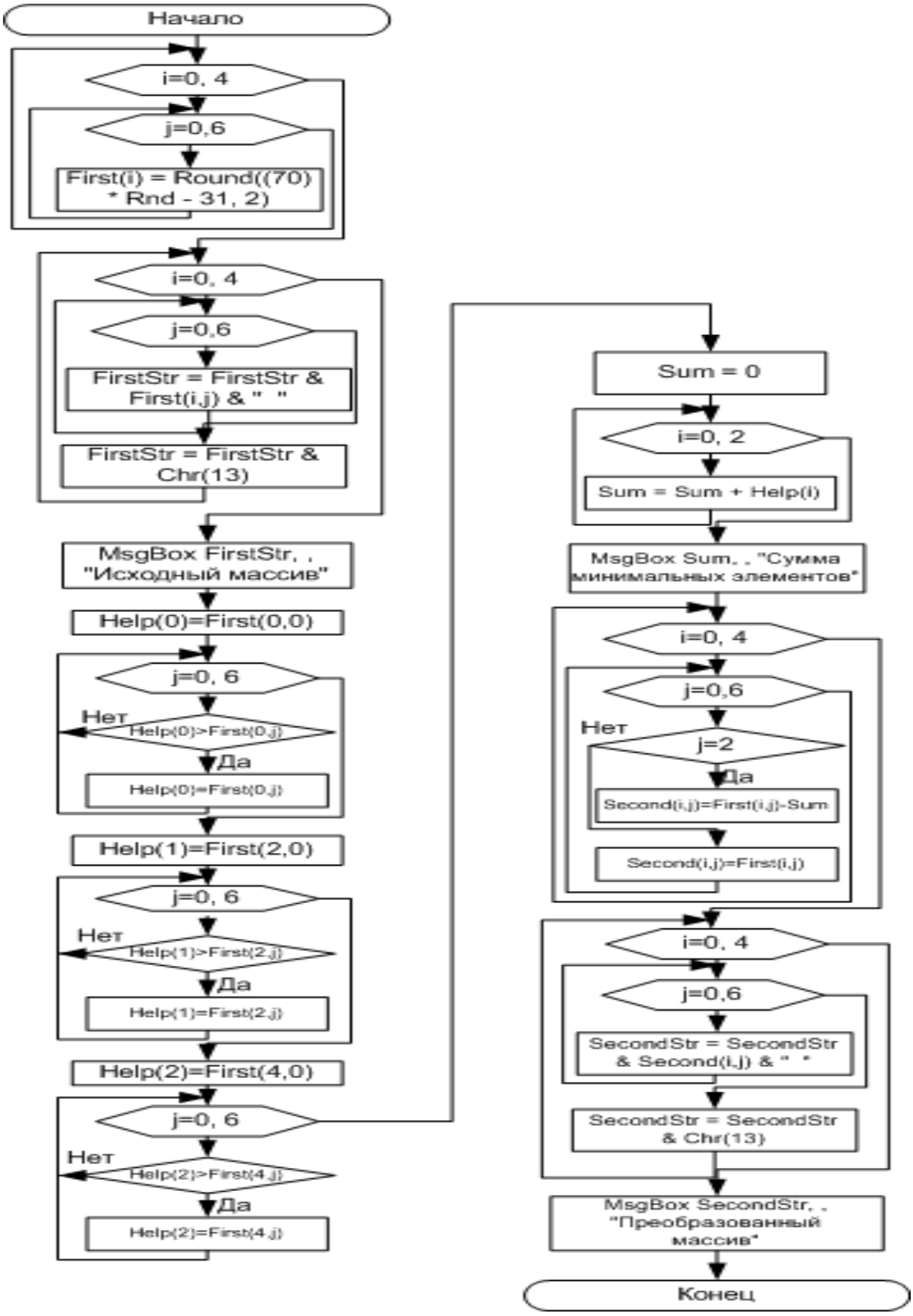
```
If Help(1) > First(2, j) Then Help(1) = First(2, j)
Next j

' Для строки 4
Help(2) = First(4, 0)
For j = 0 To 6
If Help(2) > First(4, j) Then Help(2) = First(4, j)
Next j

'Заполнить строку HelpStr элементами массива Help  и вывести ее на печать

 For i = 0 To 2
    HelpStr = HelpStr & Help(i) & "  "
Next i
MsgBox HelpStr, , "Массив минимальных элементов"

'Выполняем пункт с) Подсчитываем сумму элементов массива минимальных элементов
Sum = 0
For i = 0 To 2
'Добавляем к сумме каждый элемент исходного массива
Sum = Sum + Help(i)
Next i
MsgBox Sum, , "Сумма элементов исходного массива"

'Заполнение преобразованными элементами второго массива
For i = 0 To 4
For j = 0 To 6
If j = 2 Then
Second(i, j) = First(i, j) - Sum
Else: Second(i, j) = First(i, j)
End If
Next j
Next i

'Заполнить строку элементами массива перемежая их пробелами  и началом новой строки
'по перемене индекса i как счетчика строк и вывести на экран преобразованный массив
For i = 0 To 4
 For j = 0 To 6
SecondStr = SecondStr & Second(i, j) & "    "
Next j
SecondStr = SecondStr & Chr(13)
Next i
```

```
MsgBox SecondStr, , "Преобразованный массив"

'Вывести на экран все полученное в программе.Все вычисленное уже  выведено на экран,
'поэтому следующую команду можно не выполнять, 'но выводя на экран все одновременно
' легче проверить правильность работы всей программы

MsgBox "Исходный массив: " & Chr(13) & FirstStr & Chr(13) & _
    "Массив минимальных элементов строк 0, 2, 4:  " & HelpStr & Chr(13) & Chr(13) & _
    "Сумма минимальных элементов строк 0, 2, 4:  " & Sum & Chr(13) & Chr(13) & _
    "Преобразованный массив: " & Chr(13) & SecondStr _
     , , "Итоги работы программы"
End Sub
```

2) Таблица результатов тестирования.

Исходный First	-15.96 31.12 32.42 -5.29 -16.00 32.37 14.74 -19.67 -8.60 -9.18 -21.13 -15.18 4.16 -16.96 25.62 -6.10 2.61 11.21 20.65 -7.25 28.91 10.66 -4.35 7.19 21.54 32.86 26.18 -21.47 7.44 -23.03 22.58 -26.04 12.11 10.80 28.01
Преобразованный Second	-15.96 80.41 32.42 -5.29 -16.00 32.37 14.74 -19.67 40.69 -9.18 -21.13 -15.18 4.16 -16.96 25.62 43.19 2.61 11.21 20.65 -7.25 28.91 10.66 44.94 7.19 21.54 32.86 26.18 -21.47 7.44 26.26 22.58 -26.04 12.11 10.80 28.01
Массив Help	-16.00 -7.25 -26.04
SUM	-49.29

4) Создание блок-схемы алгоритма.


Начало
i=0, 4
j=0,6
First(i) = Round((70) * Rnd - 31, 2)
i=0, 4
j=0,6
FirstStr = FirstStr & First(i,j) & " "
FirstStr = FirstStr & Chr(13)
MsgBox FirstStr, , "Исходный массив"
Help(0)=First(0,0)
j=0, 6
Нет
Help(0)>First(0,j)
Да
Help(0)=First(0,j)
Help(1)=First(2,0)
j=0, 6
Нет
Help(1)>First(2,j)
Да
Help(1)=First(2,j)
Help(2)=First(4,0)
j=0, 6
Нет
Help(2)>First(4,j)
Да
Help(2)=First(4,j)
Sum = 0
i=0, 2
Sum = Sum + Help(i)
MsgBox Sum, , "Сумма минимальных элементов"
i=0, 4
j=0,6
Нет
j=2
Да
Second(i,j)=First(i,j)-Sum
Second(i,j)=First(i,j)
i=0, 4
j=0,6
SecondStr = SecondStr & Second(i,j) & " "
SecondStr = SecondStr & Chr(13)
MsgBox SecondStr, , "Преобразованный массив"
Конец

Лабораторная работа № 7

«Процедуры и функции»

Структуризация программ VBA

Основной структурной единицей при написании кода VBA является *процедура* (иногда называемая подпрограммой). Обычно процедура определяется как набор инструкций (операторов), направленных на решение определенной задачи. Различают два вида подпрограмм: процедуры и функции. При выполнении процедуры не происходит возвращения ни какого значения в основную программу, в то время как функция всегда возвращает определенное значение. Процедура начинается с ключевого слова Sub, а функция - Function. Синтаксис подпрограмм:

Процедура	Функция
Sub *ИмяПроцедуры*() *Код процедуры* End Sub	Function *ИмяФункции*() *Код функции* *ИмяФункции = Выражение* End Function

Если после имени подпрограммы стоят пустые скобки, значит, в подпрограмму не передаются никакие параметры, т.е. она работает без параметров.

Величина, возвращаемая функцией, может быть использована многими способами - например, как флажок, указывающий на определенное состояние, или как результат вычислений. Чтобы функция возвращала значение, *имени функции* нужно присвоить определенное выражение, как показано выше.

Обратите внимание, что имя функции представляет собой и переменную, которой должно быть присвоено значение. Если Вам не нужно, чтобы ваша подпрограмма возвращала значение, лучше использовать процедуру.

Вызов функций и процедур

Вызов процедур и функций можно осуществить из любой процедуры или функции. Например, первая из ниже приведенных процедур вызывает вторую, которая открывает окно сообщения с текстом "Всем привет".

```
Sub main()
    SecondSubroutine
End Sub

Sub SecondSubroutine()
    MsgBox "Привет всем!"
End Sub
```

При вызове процедуры (или функции) (например, из другой процедуры), управление передается вызываемой процедуре (или функции). Затем последовательно выполняются операторы вызываемой процедуры (или функции), после чего управление передается исходной (вызывающей) процедуре.

Пример использования подпрограмм и функций

Приведенная ниже процедура вызывает функцию GetLastName, аргументом которой является полное имя в форме Фамилия, Имя. Подпрограмма передает функции аргумент "Петров, Иван", а функция GetLastName использует функции VBA InStr Left$ и возвращает только фамилию.

```
Sub DisplayLastName()
    Dim StrName As String
' Вызов функции GetLastName
    StrName = GetLastName("Петров, Иван")
    MsgBox StrName
End Sub

Function GetLastName(FullName As String) As String
   Dim intLen As Integer
   IntLen = InStr(FullName, ",")
   GetLastName = Left$(FullName, intLen - 1)
End Function
```

Функция InStr определяет номер позиции, в которой находится запятая, а функция Left$ возвращает подстроку, начиная с первого символа и кончая символом, определяемым выражением intLen - 1.

1. Введите оба примера, найдите отличия между ними.
2. Запустите их на выполнение.
3. Поясните полученные результаты.
4. Задание.
5. Измените программу работы с массивами, созданную вами в работе № 5, заменив группы операторов ввода и вывода массивов процедурами, а группы операторов нахождения максимального элемента, минимального элемента и суммы массива функциями (вы должны получить не менее 2 процедур и не менее двух функций).

Пример выполнения задания

В соответствии с заданием из работы 5 нужно было ввести произвольный массив размерности 10, сделать ряд преобразований, найти сумму массива, его минимальный элемент и вывести на экран. Причем ряд действий (например вывод на экран) выполнялись несколько раз.

```
' Массив First статический с 10 элементами от 0 до 9
Dim First(9) As Integer
' Массив Second динамический
Dim Second() As Integer

Sub InputArray(Arr, Length)
'Заполнение исходного массива случайными числами
'
Randomize 'Запуск счетчика случайных чисел

For i = 0 To Length - 1
' Присвоить элементу массива случайное число от 1 до 100 со сдвигом на -20
Arr(i) = Int((100) * Rnd - 21)
Next i
End Sub

Sub ArrayToScreen(Arr, Length, Title)

'Заполнить строку элементами массива
'перемежая их пробелами и вывести на экран исходный массив

For i = 0 To Length - 1

Str = Str & Arr(i) & "  "
Next i
MsgBox Str, , Title

End Sub

Function SummaOfArray(Arr, Length)

"Подсчитываем сумму элементов исходного массива
Sum = 0
For i = 0 To 9
'Добавляем к сумме каждый элемент исходного массива
Sum = Sum + Arr(i)
Next i
SummaOfArray = Sum
End Function

Function MinOfArray(Arr, Length)

'Находим минимальный элемент
'Принимаем за минимальный элемент нулевой элемент массива
Min = Arr(0)
```

```
For i = 1 To k - 2

'Начиная с первого элемента массива проверяем, если он меньше
'минмального, то записываем его в минимальное значение
If Arr(i) < Min Then
Min = Arr(i)
End If
Next i
MinOfArray = Min

End Function


Sub Main()
' Выполняем пункт a)
' Заполнение исходного массива случайными числами и вывод его на экран
InputArray First, 10
ArrayToScreen First, 10, "Исходный массив"
'
' Выполняем пункт b)
' Подсчитать количество нечетных чисел для определения размерности массива
Second
For j = 0 To 9
   If (First(j) Mod 2) <> 0 Then
   k = k + 1
   End If
Next j
' Вывести на экран количество нечетных элементов
MsgBox k, , "Количество нечетных элементов"

' Определить размерность массива Second из k элементов от 0 до k-1
ReDim Second(k - 1)
  i = 0 'счетчик массива Second обнулить

For j = 0 To 9
  If (First(j) Mod 2) <> 0 Then

   ' Записать в массив Second нечетное число из массива First
   Second(i) = First(j)

   ' счетчик массива Second увеличивается только для нечетных чисел массива
First
   i = i + 1
   End If
Next j
```

```
' Вывести на экран преобразованный массив
ArrayToScreen Second, k, "Преобразованный массив"

' Если величина массива Second больше 3, то  выбросить из него
' элемент с номером 3
If k > 3 Then

' Начиная с элемента с номером 3 присваивать ему следующий элемент
' этого же массива
' Чтобы не выйти за пределы массива, заканчиваем на предпоследнем элементе.
   For i = 3 To k - 2
     Second(i) = Second(i + 1)
   Next i

   ' Так как количество элементов в массива Second на единицу уменьшилось
   ' изменяем его размерность на 1. Ключевое слово Preserve не позволит
   ' обнулить уже существующие элементы массива
   ReDim Preserve Second(k - 2)

' Вывести на экран сжатый массив
ArrayToScreen Second, k - 1, "Сжатый массив"
Else
' Если в массиве Second менее 4 элементов, он сжатию не подлежит
MsgBox "Массив не подлежит сжатию, т.к. его размер меньше 4", , "Сжатый массив"
End If

' Выполняем пункт c)
MsgBox SummaOfArray(First, Length), , "Сумма элементов исходного массива"
MsgBox MinOfArray(Second, Length), , "Минимальный элемент сжатого массива"
End Sub
```

Контрольные вопросы.

1. Что такое процедуры и функции и для чего они предназначены?
2. Чем они отличаются друг от друга?

Список литературы

1. Васильев А., Андреев А. VBA в Office 2000: учебный курс. – СПб.: Питер, 2001. – 432 с.: ил.
2. Гарнаев А.Ю. Excel, VBA, Internet в экономике и финансах. – СПб.: БХВ-Петербург, 2001. – 816 с.: ил.
3. Гарнаев А.Ю. MS Excel 2002: разработка приложений. – СПб.: БХВ-Петербург, 2003. – 769 с.: ил.
4. Гарнаев А.Ю. Использование MS Excel и VBA в экономике и финансах. – СПб.: БХВ-Санкт-Петербург, 1999. – 336 с.: ил.
5. Долженков В.А., Колесников Ю.В. Microsoft Excel 2000. – СПб.: БХВ-Петербург, 1999. – 1088 с.: ил.
6. М. Коттингхэм Excel 2000: руководство разработчика: Пер. с англ. – К.: Издательская группа BHV, 2000. – 704 с.
7. Матросов А.В. и др. MS Office XP: разработка приложений / Матросов А.В., Новиков Ф.А., Усаров Г.Е., Харитонова И.А. / Под ред. Ф.А. Новикова. – СПб.: БХВ-Петербург, 2003. – 944 с.: ил.
8. Саймон, Джинжер Программировагние в Excel: наглядный курс создания интерактивных электронных таблиц. : Пер. с англ. – М. : Издательский дом «Вильямс», 2002. – 336 с.: ил.
9. Самоучитель программирования на VBA в Microsoft Office / Хорев В.Д. – К.: Юниор, 2001. – 296 с., ил.
10. Харрис, Метью Освой самостоятельно программирование для Microsoft Excel 2000 за 21 день.: Пер. с англ. : Уч. пос. – М.: Издательский дом “Вильямс”, 2000. – 880 с.: ил.

УЧЕБНОЕ ИЗДАНИЕ

Методические указания для выполнения лабораторных работ по курсу «Экономическая информатика» (для студентов 1, 2 курсов дневной формы обучения бакалавров направления 6.030504 - «Экономика предприятий», 6.030509 – «Учет и аудит»).

Составители: Борис Иванович Погребняк, Анна Викторовна Белогурова, Екатерина Владимировна Кузьмичева, Наталья Олеговна Манакова.

Редактор: Н. З. Алябьев

План 2008 , поз. 171 М

Подп. в печать 25.03.08.	Формат 60х84 1/16	Бумага офисная.
Печать на ризографе.	Условн.-печ. л. 3,8.	Уч.- изд. л. 4,0
Тираж 50 экз.	Зак. № .	

61002, Харьков, ХНАГХ, ул. Революции, 12

Сектор оперативной полиграфии ИВЦ ХНАГХ

61002, Харьков, ул. Революции, 12